A GENETIC ALGORITHM APPROACH TO SOLVE A FLEXIBLE JOB SHOP PROBLEM WITH DUE DATE

WALEED AHMED

MS-DME-00000119966

ADVISOR

DR SHAHID IKRAMULLAH

DEPARTMENT OF DESING AND MANUFACTURING ENGINEERING

SCHOOL OF MECHANICAL AND MANUFACTURING ENGINEERING

NATIONAL UNIVERSITY OF SCIENCES AND TECHNOLOGY

2017

A Genetic Algorithm Approach to Solve Flexible Job Shop Problem with Due Date

WALEED AHMED

00000119966

In partial fulfillment of the requirements for the degree of

MS Design and Manufacturing Engineering

Thesis Supervisor:

Dr. Shahid Ikramullah

Thesis Supervisor's Signature: _____

Department of Design & Manufacturing Engineering
School of Mechanical & Manufacturing Engineering (SMME)

National University Of Sciences and Technology

H-12, Islamabad

Thesis Acceptance Certificate

Certified that final copy of MS/MPhil thesis written by **Mr. Waleed Ahmed**, (Reg. No. **00000119966**), of **SMME** (School/College/Institute) has been vetted by undersigned, found complete in all respects as per NUST Statutes/Regulations, is free of plagiarism, errors and mistakes and is accepted as partial fulfilment for award of MS/MPhil Degree. It is further certified that necessary amendments as pointed out by GEC members of the scholar have also been incorporated in the said thesis.

Signature: _____

Name of Supervisor: **Dr. Shahid Ikramullah Butt**

Date: _____

Signature (HOD): _____

Date: _____

Signature (Principal): _____

Date: _____

FORM TH-4

National University of Sciences & Technology

MASTER THESIS WORK

We hereby recommend that the dissertation prepared under our supervision by: Waleed Ahmed,Reg # 00000119966 Titled:

"*A Genetic Algorithm Approach to solve a Flexible Job Shop Problem With due date"* be accepted in partial fulfillment of the requirements for the award of Masters of Science in Design and

Manufacturing Engineering Degree with (grade)

Examination Committee Members

1. Name: Dr. Abdul Gafoor Signature: _____

2. Name: Dr. Liaquat Ali Signature: _____

3. Name: Dr. Hussain Imran Signature: _____

Supervisor's name: Dr.Shahid Ikramullah Signature: _____
Date: __/__/2017

_____ _____

Head of Department Date

COUNTERSINGED

_____
Dean/Principal

Date: _____

DECLARATION

I certify that this research work titled "*A Genetic Algorithm Approach to solve a Flexible Job Problem with due date."* is my own work. The work has not been presented elsewhere for assessment. The material that has been used from other sources it has been properly acknowledged / referred.

Waleed Ahmed

00000119966

CERTIFICATE FOR PLAGIARISM

It is certified that PhD/M.Phil/MS Thesis Titled ***A Genetic Algorithm Approach to solve a Flexible Job Shop Problem with due date*** by **Waleed Ahmed** has been examined by us. We undertake the follows:

a. Thesis has significant new work/knowledge as compared already published or are under consideration to be published elsewhere. No sentence, equation, diagram, table, paragraph or section has been copied verbatim from previous work unless it is placed under quotation marks and duly referenced.

b. The work presented is original and own work of the author (i.e. there is no plagiarism). No ideas, processes, results or words of others have been presented as Author own work.

c. There is no fabrication of data or results which have been compiled/analyzed.

d. There is no falsification by manipulating research materials, equipment or processes, or changing or omitting data or results such that the research is not accurately represented in the research record.

e. The thesis has been checked using TURNITIN (copy of originality report attached) and found within limits as per HEC plagiarism Policy and instructions issued from time to time.

Name & Signature of Supervisor

Signature: _____

COPYRIGHT STATEMENT

a. Copyright in text of this thesis rests with the student author. Copies (by any process) either in full, or of extracts, may be only in accordance with the instructions given by author and lodged in the Library of SMME, NUST. Details may be obtained by the librarian. This page must be part of any such copies made. Further copies (by any process) of copies made in accordance with such instructions may not be made without the permission (in writing) of the author.

b. The ownership of any intellectual property rights which may be described in this thesis is vested in SMME, NUST, subject to any prior agreement to the contrary, and may not be made available for use of third parties without the written permission of SMME, NUST which will describe the terms and conditions of any such agreement.

c. Further information on the conditions under which disclosure and exploitation may take place is available from the library of SMME, NUST, Islamabad

ACKNOWLEDGEMENTS

In the name of Allah, the Magnificent, the Merciful.

Research work is a never ending struggle, and I am happy to have support from family and friends. I have my fair share of mistakes along the way, but as Einstein once said, “Anyone who has never made a mistake has never tried anything new.”

After each road block, Dr. Shahid Ikramullah has been a beacon of hope, enlightening the path; guiding. I am forever in gratitude to Dr. Shahid Ikramullah, my research supervisor also the encouragement and guidance of my Guidance and Examination Committee is estimable.

I am profusely thankful to my parents and siblings who have continuous support for me throughout in every department of my life. I also owe my deep gratitude to my fellows as their constructive criticism and support helped me a lot during this journey.

To everyone, who were a part of this journey with me, thank you and wish you all the best in future.

TABLE OF CONTENTS

LIST OF FIGURES:

LIST OF TABLES:

ABSTRACT

Scheduling is one of the most critical elements of any production system as many shop activities are basing on it and the optimization of system performance procedures can be done with proper scheduling of shop floor activities. Flexible Job Shop Scheduling Problem (FJSSP) is a vital scheduling problem which has received substantial significance in the manufacturing area. Flexible Job-shop Scheduling Problem is the extension of Job Shop Scheduling (JSS) which allows an operation to be executed on more than one machine and is also computationally NP-hard. Our study proposes an efficient scheduling method using the genetic algorithm (GA) along with the due date in order to minimize the "make-span time" (C_{max}) as it is one of the imperative parameter of such problem. This research will discuss the outcomes of the FT06 problem concerning the makespan. And in the end this dissertation will draw deduction on the basis of this research and will provide some recommendations/proposal as well. The results of the proposed strategy prove it to be a more resourceful technique as compared to the existing methods.

CHAPTER 1

INTRODUCTION

1. BACKGROUND

In modern era manufacturing industries have to face many challenges and the competition among them is very high. This tough competition advises maximization of profit and minimization of wastes. Here comes the role of optimization which provides optimal solution to these problems in minimum time duration. Optimization commonly involves mathematical modeling of the given problem, selection of the most appropriate solution technique and knowledge of programming language.

Production industries are the pillar in the financial structure of any nation as they subsidize to both growing Gross Domestic Product and creating recruitment opportunities. Production efficiency can be maximized by utilizing the existing resources in an optimum manner and the only possible way to optimized utilization of resources is by using proper scheduling system. This makes scheduling very significant feature of a manufacturing system.

1.1.Processing Layouts

Scheduling theory has been developed to address various types of machine layouts.

1.1.1. Flow Shop Layout

It is a form of facility setup where all the jobs follow the same processing orders on all the machines and this type of layout is also called Line Layout. This type of layout is mostly used for mass production where job flow is fixed. One of the advantage of using this facility setup is that, it is easier to control the flow of jobs in it, while one of the drawback of using this layout is that responding to the diversification of product is quite difficult.

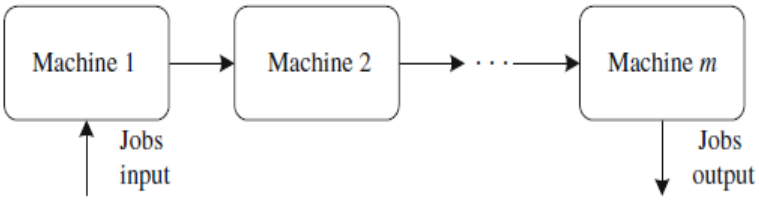


Figure 1: Flow Shop Layout

1.1.2. Open Shop Layout

In open shop jobs must be processed for certain amount of time at each sets of defined work centers irrespective of the proper sequence. No pre-determined sequence is followed therefore jobs can be handled in any sequence.

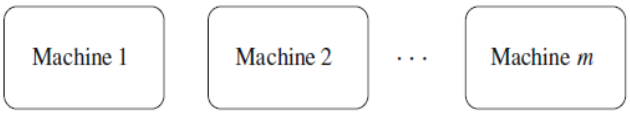


Figure 2: Open shop layout

1.1.3. Job Shop Layout

A job shop is a type of facility setup in which small lots of a variety of custom products are produce. In a Job-Shop every job may have a distinct processing sequence. In this products require a distinctive sequencing of process steps. One of the advantage using this type of facility is that it provides high flexibility while at the same time it is quite difficult to control because of high product variability. Some of the models of job shops include a machine tool shop, a machining center, a coat shop, printing shop, and other producers that make custom items in small batches.

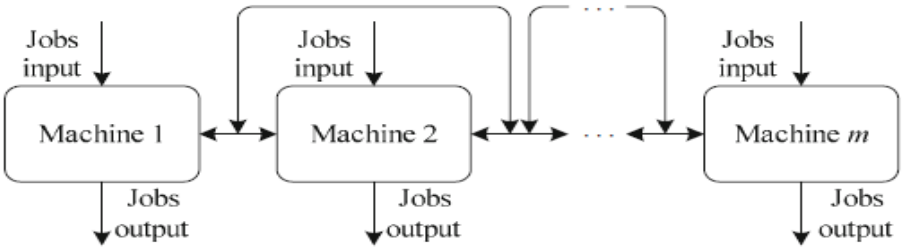


Figure 3: Job shop layout

1.1.4. Mixed Shop Layout

Mixed shop is mainly the mixture of flow-shop and Job-Shop. In this type of facility setup few jobs have permanent machine sequence like in a flow-shop, and few are handled in a random sequence like a Job-Shop. Models described above are truly the byproducts of open shop layout.

1.2.Scheduling

Scheduling can be described as, "it is a function to determine an actual (optimal or feasible) implementation plan as to the time schedule for all jobs to be executed; that is, when, with what machine, and who does what operation" Hitomi [1]. Wiers [2] suggests that production scheduling is the performance of operations on set of jobs, with the help of previously assigned set of machines within a definite time. Another definition of scheduling is that, "the allocation of resources over a period of time to perform a collection of tasks" Noor [3]. Scheduling has its application universally, for example; production and procurement, transportation and distribution, information processing and communications etc. Scheduling has also been applied to fulfill customer request, which plays an significant role in customer satisfaction, "Scheduling" and "Sequencing" are usually considered as a same. Conway, Maxwell, and Miller [4] stated that sequencing problem occurs when there is an order in the number of tasks that must be performed.

According to the class of activities, scheduling can be generally categorized into project scheduling and operation scheduling.

Project Scheduling is the scheduling of different activities involved in executing a project. It can also be defined as the discipline for guiding how to carry out the project within a definite timespan, usually with well-defined stages and with nominated available resources. Effective project scheduling or planning guarantees that the system are delivered with in cost, time and quality constraints. Project can be construction of an industry, maintenance of a bridge or a factory shop etc. Some renowned techniques to handle such kind of scheduling are; Critical Path Method (CPM), Gantt chart, Schedule Network Analysis and Project Evaluation and Review Technique (PERT).

Operation scheduling includes assigning jobs to workstations or personnel to jobs for definite time periods. Hitomi [1] defines operation scheduling as, "the processing of a set of jobs, in a given amount of time, on the already allocated corresponding set of machines.

Jain [5] categorized the existing operations scheduling layout as job sequencing, flow-shop scheduling, mixed-shop scheduling, and job-Shop scheduling and open-shop scheduling.

Job sequencing model defines the sequence in which a set of jobs would be processed on one machine. For N jobs there are a set of N! Number of possible sequences. From these N! Number of sequences, one sequence is selected based on the minimization or maximization of defined objective functions.

Flow shop scheduling all the jobs follow the same processing orders on all the machines. This type of scheduling is mostly used for mass production. In flow job scheduling there is a strict order of operations to be performed on all the jobs. The job can be processed on tops one machine; meanwhile one machine can process tops one job.

Job-Shop Scheduling each job may have a distinct handling sequence. "Job-Shop has a typical arrangement for the case of varied production of most jobbing types and batch types" Hitomi [1]. Scheduling of Job-Shop is more complex as related to the scheduling of flow-shop. As each job has a distinct processing sequence, thus for every machine a different job sequence has to be defined. Job orders should be inter-related with each other in order to achieve minimum make span time.

Mixed shop scheduling fundamentally the mixture of flow-shop scheduling and Job-Shop scheduling. In this type of scheduling few jobs have stationary machine sequence like in a flow-shop, and few are processed in a random sequence like a Job-Shop. In other words, "jobs must be processed in a sequence consistent with a given partial order of machines in mixed shop" (Jain [5]).

Flexible Job Shop Scheduling: The FJSSP is a further extension of JSSP in which the operations can be performed on any machines which can be selected from a finite number of given set of machines in a flexible manufacturing cell. Thus the problem is intricate in a sense that it also involves machine assignment problem for each operation and thus it is subdivided into following two parts:-

a. Routing; i.e. through which machines the jobs should be processed from an available set of machines
b. Sequencing; i.e. in which order the jobs should be processed on the selected machines

Thus there is inherent "flexibility" in the FJSSP in contrast to the JSSP. Flexibility has been introduced in the classical JSSP in some of the following ways:-

The idea of FJSSP was first adopted by Brucker et al [6] as multi-purpose machines equipped with different tools.

a. Barnes et al [7] argued that a JSSP can be converted into FJSSP by incorporating multiple instances of a single machine where a bottleneck is encountered during the scheduling process.
b. Najid et al [8] argue that flexibility is bought in the JSSP with the condition that one machine may be able to perform more than one type of operation.

Kacem et al [9] classifies the FJSSP into following types:-

a. Total FJSSP (T-FJSSP): In this type, required operation can be performed on any of the available machines in the shop, thus complete flexibility has been achieved;
b. Partial FJSSP (P-FJSSP): In this type, operations can only be performed on specific machines and remaining operations can be executed on any of the machines in the shop.

According to Chan et al [10], there are following 2 types of FJSSP:-

a. Type I FJSSP: In this type jobs under consideration have different operation sequences and identical / non-identical machines for every operation. In this problem, the interest is to find the operation sequence and job processing order.
b. Type II FJSSP: In this type jobs under consideration have fixed operation sequences, nonetheless different identical or non-identical machines for each and every operation. In this problem, the interest is to allocate jobs to machines according to their operation sequences.

1.3. Scheduling Algorithms

Algorithms are computer programs that are used to achieve solution of a predefined mathematical problem by repeating itself up to a desired point. Or it can also be defined as the step by step technique for solving a pre-determined problem using computer. Algorithms have the capability of calculating, data processing and automated reasoning.

These algorithms can be classified as:

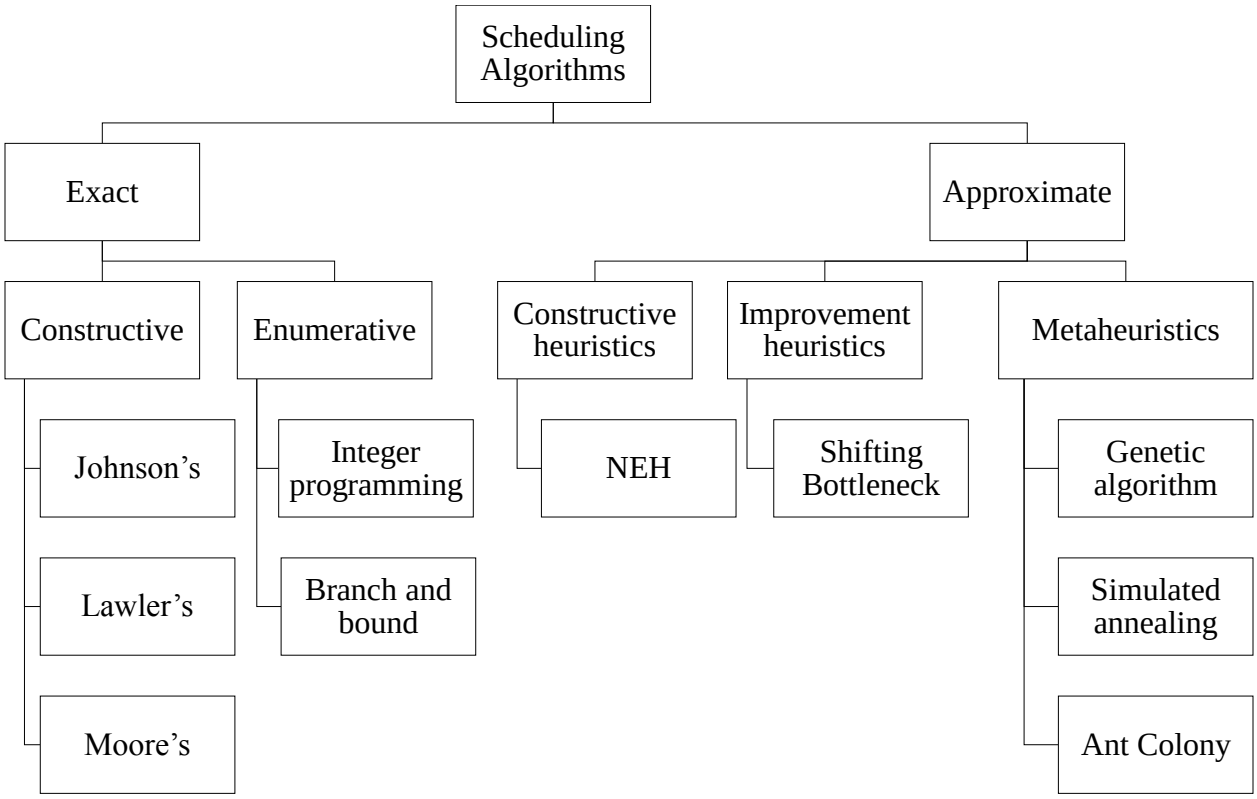


Figure 4: Classification of scheduling algorithms

1.3.1. Exact algorithms

The schedule obtained after the iteration from exact algorithm is the best and no other schedule can perform better.

1.3.1.1. Exact constructive algorithms

These algorithms utilizes few properties of the specific model in order to build a solution which is certain to be optimal, e.g., Johnson's algorithm, Lawler's algorithm and Moore's algorithm.

1.3.1.2. Enumerative algorithm

These algorithms directly or indirectly guarantee to estimate all possible solutions of the model. It comprises complete enumeration as well as branch and bound, cutting plane, or branch and cut approaches, e.g., Integer programming, Problem oriented branch and bound approaches.

1.3.2. Approximate algorithms

The schedule obtained after the iteration from approximate algorithm is not guaranteed to be best and experience is required to assess the same.

1.3.2.1. Constructive heuristics

Constructive heuristics in manufacturing scheduling produce a optimistically feasible and good sequence/schedule from the beginning, i.e. exclusively based on the input statistics of the problem, without mentioning to another solution to the problem under consideration, e.g., the NEH heuristic.

1.3.2.2. Improvement heuristics

Improvement heuristics try to develop a improved solution based on one or more solutions found so far. 'Better' here in the extensive majority of scientific contributions mentions to the enhancement of the objective function value of solutions but might also discuss the lessening of infeasibility, or both, e.g., The Shifting Bottleneck Heuristic.

1.3.2.3. Metaheuristics

Because of the non-convexity of several combinatorial optimization problems and its results of easily getting trapped in local sub-optima when applying constructive heuristics in combination with deterministic 'permanent improvement' neighborhood search approaches, during the last periods numerous methods have been established which try to overcome or at least decrease this problem. These approaches methodically try to enlarge the search space by letting temporal and stochastically deteriorating of solutions to be considered for the next neighborhood search and/or to adjust philosophies from phenomena of nature to explore also rather different solutions which

are omitted by pure neighborhood search, e.g., Descent search methods, Simulated annealing, Tabu search, GA, ACO etc.

1.4.Evolutionary Algorithms

Evolutionary algorithms (EAs) are search procedures that take their motivation from natural selection and survival of the fittest in the natural. Evolutionary algorithms vary from more traditional optimization methods in that they involve a search from a "population" of solutions, not from a solo point. Every iteration of an EA comprises of a competitive selection that discards poor solutions. A basic structure of evolutionary algorithms is shown in Figure5 below.

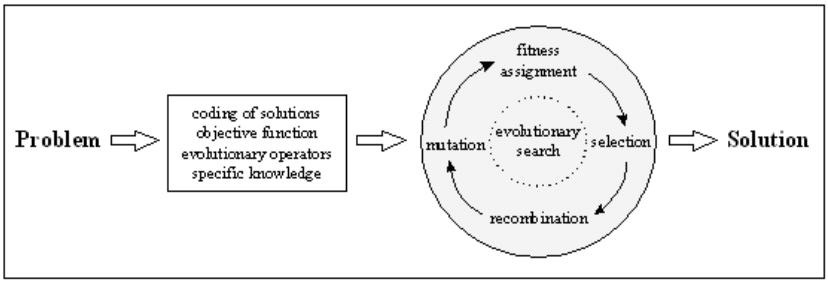


Figure 5: Problem solution using evolutionary algorithms

Different categories evolutionary algorithms have evolved during the last 40 to 50 years.

Some of include:

1) Genetic Programming (GP), which develop programs and are well matched for problems that involve the determination of a function that can be basically uttered in a function form.

2) Evolutionary Programming (EP), which emphases on improving incessant functions without recombination.

3) Evolutionary Strategies (ES), which concentrates on improving continuous functions with recombination developed in Germany by I. Rechenberg

4) Genetic Algorithms (GAs), which emphases on enhancing general combinatorial problems mainly developed in the USA by J. H. Holland

CHAPTER 2

LITERATURE REVIEW

2. INTRODUCTION TO GENETIC ALGORITHM

Genetic algorithms belong to the evolutionary algorithms class and its development was inspired through the process of natural genetic evolution. The original work on natural evolution was contributed by Darwin [11] in which he claimed that natural populations evolve according to the process of natural selection on the basis of "survival of the fittest" rule. Initial work on GA was conducted by Holland [12] in 1975, which was then extended majorly by Goldberg [13].

Giraffes use their long necks eat the leaves at higher parts of the plants. Thus as per the rule of the survival of the fittest, giraffes have evolved with generations having longer necks. The GAs can be used to mimic this natural process of genetic evolution on the principal of survival of the fittest to obtain solutions to the engineering problems. The beauty of GA lies in its 'adaptive' nature; i.e. it can change / fit itself according to the changing environment. Next section explains basic working of GA.

The basic working element of GA is gene, a group of which constitutes a chromosome. The chromosomes contain the current state data coded in the form of binary digits 0 or 1. This structure represents a candidate solution to the problem in consideration. GA works on these coded forms of the data instead of working on actual data elements. The chromosomes are also referred to as individuals and they combine to form a population. A number of populations altogether formulate a generation over each iteration. Figure represents the schematic representation of the relation of these elements.

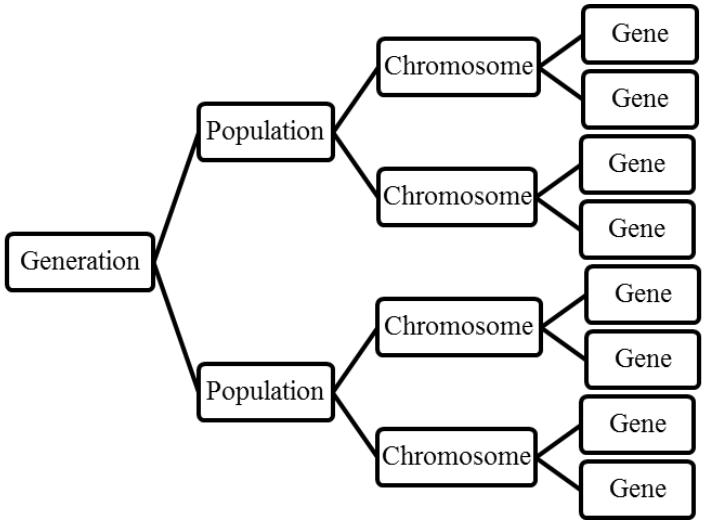


Figure 6: Basic Elements of GA

2.1.Fundamental Models

Genetic Algorithms use a language rented from natural genetics. Set of individuals are called population. There are 2 representations for an individual called genotype and phenotype .Following table gives a list of different expressions, which are most common in genetics. [14]

Table 1: Contrast of Natural Evolution with Genetic Algorithm

Natural Evolution	Genetic Algorithm
Phenotype	Uncoded point
Genotype	Coded string
Gene	String position
Fitness	Objective function value
Allele	Value at a certain position
Chromosome	String

The phenotype signifies a possible solution to the problem to be improved in a upfront way used in the original formulation of the problem. The genotype, instead, gives an "encoded" representation of a possible solution in the form of a "chromosome". A chromosome is compromises of genes, which is a set of codes consisting of a string of characters that are organized in linear sequence and every gene controls the inheritance of one or numerous characters or features. Chromosomes are also called individuals in the literature. E.g, a chromosome may consist of a sequence of 0 or 1 (i.e. a bit string), and the value at a sure position relates to on (the value = 1) or off (the value = 0) of a certain feature. More complex forms such as a sequence of symbols and a permutation of alphabets are selected for chromosomes depending upon the target problem. Every individual has its fitness, which measures how appropriate is the individual for the local environment. Darwinian theory says that among individuals in a population, the one that is the most appropriate for the local environment is almost certainly to survive to have greater numbers of off spring. And also called a rule of "survival of the fittest." Objective function f of the target optimization problem plays the role of an environment, thus, the fitness of an individual F tells us how "decent" is the matching potential solution in terms of the original optimization standards. When the target optimization is the maximization of the objective function, then fitness may be identical to the objective function value:

$$F(x) = f(x)$$

here x is the individual in the existing population P (Set of individuals or possible solutions). But in case the target is the minimization, at that point the objective function need to be changed so that an individual with a lesser objective function value has a greater fitness.

2.2.A Simple GA

First of all, the problem has to be coded in such a way that it can be represented in the form of binary codes in a chromosome. The initial solution can be generated by randomization or diversification. Then, to initiate a GA, a set of initial candidate solutions are required. The initial solution is then subjected to genetic operators (selection, crossover, mutation) until the termination criteria are met. Figure represents a typical GA [15]

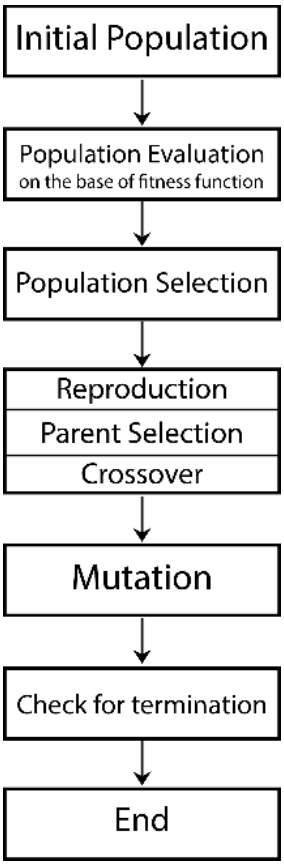


Figure 7: Simple GA search process

2.2.1. Genetic Operators

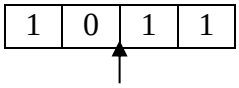
As indicated before, a number of populations form a generation. The generation is then subjected to the genetic operators to obtain a new generation. The new generation is theoretically better than the previous generation, as the new generation is generated after implementing the principle of "survival of the fittest". New generations are formulated by replacing the older generations. During this process, either the whole population can be changed or only the worst chromosome can be replaced. Obviously, these are two extreme methods and several strategies for new population can be formulated. The iterations are guided in a way that they satisfy a fitness criteria, thus the iterations are repeated to obtain an acceptable generation. These operators are basically used to

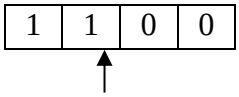
bring in the beauty of randomization in the algorithm. Standard GA operators are presented in the following:

Selection: Selection operator is used on chromosomes in a generation for reproduction and thus new generations are produced. The chromosomes satisfying the fitness criteria are likely to be selected in each newer generation. Generally used selection criteria are:-

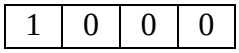
i. *Roulette wheel selection*: The selection probability of a chromosome is directly proportional to its fitness as assessed by the fitness criteria.
ii. *Rank based fitness assignment*: This method associates relative fitness between individual chromosomes, hence preventing a generation to contain an all-fit chromosome structure. The method is mainly used to maintain diversity in the population.
iii. *Tournament selection*: Set of chromosomes are chosen in a random way and then the fittest chromosomes are selected for further operation. This method is completely random.
iv. *Elitism*: The crux of this method is that it maintains a fixed number of fittest chromosomes and the rest of the population is generated by using any of the preferred selection methods. Thus this method not only ensures that the best solutions remain in the population, but also ensue the diversification of the population by selecting chromosomes from the entire solution space.

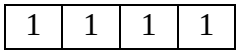
Crossover: The crossover operator is applied on the genes of two chromosomes to produce two off springs which contain distinctiveness of the parent chromosomes, and thus these off springs have more probability of survival than their parents. Consider two chromosomes having 4 genes each as follows:-





Crossover can be applied to these chromosomes at third gene (at pointed arrows) to obtain following two offsprings:-





Now these are two new chromosomes having genes of two previous chromosomes. The technique elaborated above is known as single point crossover. Many modified

crossover techniques have been proposed in literature which will be identified in this review.

Mutation: Mutation operator is applied on a single chromosome for the purpose of changing a gene at its respective location. The gene 1011 can be mutated as 1111, as the gene at location 2 is flipped from 0 to 1. The mutation operator is used to change some information in a selected chromosome or diversify the solution space for further exploration. Many modified mutation techniques have been proposed in literature which will be identified in this review

2.3. Introduction to Scheduling Rules:

Model of Job sequencing defines the sequence where a set of jobs would be managed on one machine. For N no of jobs there are a set of N! Which represents No. of possible sequences. From these N! no. of sequences, only 1 sequence is selected based on the minimization or maximization of well-defined objective functions.

There are some priority rules that provide guidelines for the sequencing of jobs. Some of the commonly used priority rules are:

FCFS (First Come First Serve): Jobs are processed on the basis of their arrival at the work center, job that arrive first at the work center is processed first. The objective of FCFS approach is to reduce the completion time for that exact jobs. This type of scheduling is suitable for service organizations (Fogarty et al. [1991[17]) and (Vonderembse and White [18]). A major drawback of this method is that it does not provide constant results. Additional shortcoming is that no other constraint is considered other than the arrival.

SPT (Short Processing Time): In this type of approach job with shortest processing time is processed first. Jobs are sequenced based on their processing time . Objective for SPT approach is the decrease in the mean work in process, mean job completion and also mean job lateness (Smith 1989[16], Fogarty et al. [1991] and Vonderembse and White [1991]). Although results from these rules are continuous but still one of the drawback of this approach is that job having lengthiest processing time is processed at the end.

EDD (Earliest Due Date): Jobs are processed on the basis of their due date, job with earliest due date is processed first at the work center or machine. The objectives for EDD approach are minimization of maximum lateness and mean tardiness. One of the drawback of this approach was observed by Vonderembse and White [1991] that since only one job is processed at a time, which results in missing other jobs due dates.

LPT (Longest Processing Time): Similar to SPT, jobs are processed on the basis of their processing time but in LPT jobs with longest processing are processed first.

CR (Critical Ratio): This is the ration between the time residual until due date to the total shop time residual.

$$\text{Critical Ratio} = \frac{\text{Due date - Today's date}}{\text{Total shop time remaining}}$$

The objectives for this approach are minimization of lateness and tardiness. Job having minimum critical ratio is arranged first and so on.

2.4.Scheduling Algorithm - Example Problem

Let us consider an example problem with the following specifications:

No of Jobs = Ji; i =1 to 4

No of Work Centers = Mm; m = A, B, C, D

Operations of Job 1 = O1j; j= 1 to 4

Operations of Job 2 = O2j; j= 1 to 4

Operations of Job 3 = O3j; j= 1 to 4

Operations of Job 4 = O4j; j= 1 to 4

Table 2: Shows the Processing time

Jobs	Job Sequence μ(Oij)	Processing Times (in Min) p_{ijk}	Due Date (in Min)
1	A, B, C, D	5, 3, 7, 2	20
2	B, D, C, A	8, 6, 2, 5	30
3	B, D, C, A	3, 6, 4, 4	18
4	A, C, B, D	2, 2, 7, 3	15

Let us assume that work centers A, B, C and D consists of two identical machines in each work center (Table 4-5). Table 4-6 represents operations for each job. Critical path diagram is also shown in Figure 4-10.

Identical machines in work center A = A1, A2

Identical machines in work center B = B1, B2

Identical machines in work center C = C1, C2

Identical machines in work center D = D1, D2

Table 3: Shows the Work centers

Machines	Machine A		Machine B		Machine C		Machine D	
identical machines in work centers	A1	A2	B1	B2	C1	C2	D1	D2
Real number allocation	1	2	3	4	5	6	7	8

Table 4: Demonstrates the Processing Sequence

No. of jobs	J 1	J 2	J 3	J 4
operations of each job	J1-1, J1-2, J1-3, J1-4	J2-1, J2-2, J2-3, J2-4	J3-1, J3-2, J3-3, J3-4	J4-1, J4-2, J4-3, J4-4

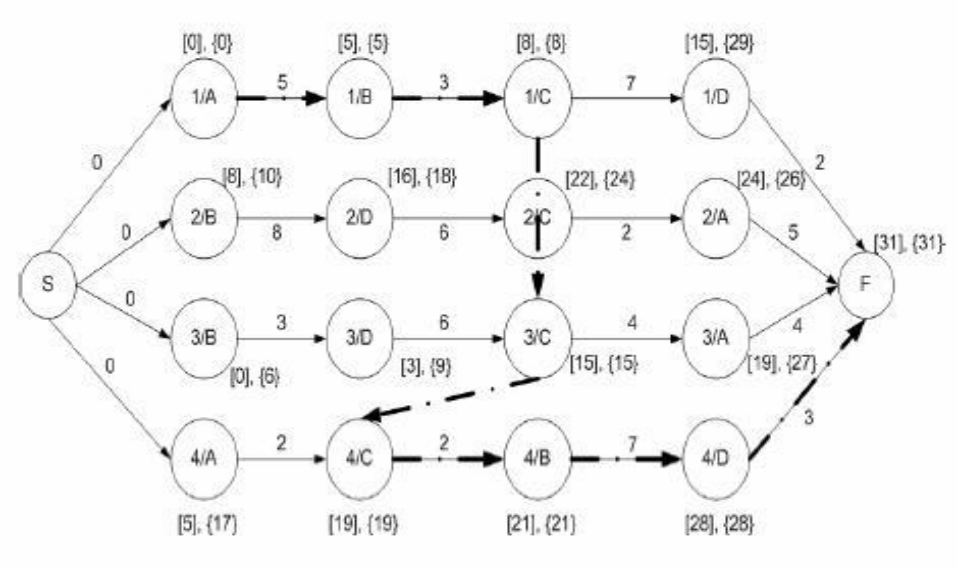


Figure 8: Critical path diagram for the example

As we know, FJSS is a special case of JSS (Job Shop Scheduling problem) [19] . Not like Job Shop Scheduling, Flexible Job Shop has more than single work centers where jobs can be processed. A exact operation of a job can be processed by the work center and in this approach any machine included in that work center can do that operation. Problem or the drawback of this technique is to allocate operation to a work center and then to any machine in a way that Makespan (maximum completion time of all operations) is minimized. In the presented approach, the work centers are assigned operations of the jobs, in such a way that each gene of the chromosome signifies the machine no. according to the assignment of the job operations to them. For each gene signifies a real number for the representation and binary representation is not taken into account as it can produce difficulty during encoding and decoding process.

Availability and job routing are the chosen criteria for the machines in each work center. In this manner, the jobs 1, 2, 3, 4 are encoded till the operations of all jobs are checked in turn. Field of the gene is limited within the Work Center that can process current operation. Sample chromosomes for example problem are shown in.

Table 5: Shows the Sample Chromosomes

Sample Chromosome 1	1, 3, 5, 8, 4, 7, 5, 1, 3, 7, 6, 2, 2, 6, 4, 8
Sample Chromosome 2	1, 4, 6, 7, 3, 8, 6, 1, 3, 7, 5, 1, 1, 6, 3, 8

The length of chromosome is grounded on the no. of work centers and the no. of jobs.

No of Work centers = 4

No of Jobs = 4

Chromosome length = 4x4 = 16

After the generation of population, fitness of each chromosome is calculated according to the population size. The population size can be chosen as 30, 50 or 100. Initially a population size of '30' and the generation size of '30' are selected respectively. Each chromosome consists of genes, which represent the process assigned to each machine, and the scheduling of processes on each machine is done with the rules like SPT and LPT.

The sample chromosome represents the machines within the work centers. For example, 1, 3, 5, 8 represents the allocation of processes Job1-1, Job1-2, Job1-3, Job1-4 to machines with in respective work centers and so on. This process continues for all the chromosomes in the population.

Evaluation of the chromosome

Evaluation of chromosome is an important part of the genetic algorithm. Although there are different methods to evaluate a chromosome but I have used Makespan as the fitness function to evaluate a chromosome. A sample chromosome is interpreted as a complete schedule shown in figure 4-8 and the schedule has a make span of 21 time units. Thus, the value of the genetic algorithm's fitness value is 21.

Scheduling Rules

After the generation of chromosome, scheduling rule (EDD) are used to schedule each process assigned to each machine with in the work center. Fitness of the chromosome is found out when all the processes are assigned to all the machines and then the schedule is generated in the form of a GANTT CHART after the application of Crossover, Mutation and Selection operators.

Crossover

Two parent chromosomes are selected in a random way from the sample of a particular population size (e.g. 50 or 100) for a two point crossover. Furthermore, all the Crossover Chromosomes stay within the population.

Table 6: Shows Sample Parent Chromosomes

Indexing	0, 1, 2, 3, 4, 5, 6, 7, 8, 9,10,11,12,13,14,15
Sample Parent Chromosome 1	1, 3, 5, 8, 4, *7, 5, 1, 3, 7, 6, 2, 2, 6,* 4, 8
Sample Parent Chromosome 2	1, 4, 6, 7, 3, *8, 6, 1, 3, 7, 5, 1, 1, 6,* 3, 8

Let;

Position 1 = 5

Position 2 = 14

Table 7: Shows the Sample Parent Chromosomes

Indexing	0, 1, 2, 3, 4, 5, 6, 7, 8, 9,10,11,12,13,14,15
Sample Child Chromosome 1	1, 3, 5, 8, 4, *8, 6, 1, 3, 7, 5, 1, 1, 6,* 4, 8
Sample Child Chromosome 2	1, 4, 6, 7, 3, *7, 5, 1, 3, 7, 6, 2, 2, 6,* 3, 8

Pairs of individuals selected go through crossover with probability *P*c. Random no. *R* is produced in the range 0-1 and the individual undergo crossover if and only if *R*<*P*c then crossover process is executed as follows: [20]

$$Pc = \begin{cases} K1 & f1 > fav \\ K2 & f1 < fav \end{cases}$$

Where f_1 is the greater fitness value of the two parents and f_{av} is the mean or the avg fitness value of the population. Different values of K_1 and K_2 can be used to find out a near optimal value of the Makespan.

This choice of crossover rate is open but DeJong [21] and Grefenstette [22] have introduced some guidelines for the selection of crossover rate as well as mutation rate.

Crossover rate = 0.6 (For large population size; 100)

Crossover rate = 0.9 (For small population size; 30)

Based on this guideline Crossover rates K_1 and K_2 were selected as follows:

Crossover Rate K_1 = .90

Crossover Rate K_2 = .85

Mutation

After the calculation of fitness and avg values of all the individuals in the population with new Child Chromosomes, which now, become part of the population, mutation operation will take place. Mutation position selected = 12 Table 8 shows the Sample parent chromosome for mutation

Table 8: Shows the Sample Parent Chromosomes for Mutation

Indexing	0, 1, 2, 3, 4, 5, 6, 7, 8, 9,10,11,12,13,14,15
Sample parent Chromosome 1	1, 3, 5, 8, 4, *8, 6, 1, 3, 7, 5, 1, 1, 6,* 4, 8
After Mutation Chromosome 1	1, 3, 5, 8, 4, *8, 6, 1, 3, 7, 5, 1, 2, 6,* 4, 8

Mutation position selected = 6

Table 9: Shows the Sample Parent Chromosomes for Mutation

Indexing	0, 1, 2, 3, 4, 5, 6, 7, 8, 9,10,11,12,13,14,15
Sample parent Chromosome 2	1, 4, 6, 7, 3, *7, 5, 1, 3, 7, 6, 2, 2, 6,* 3, 8
After mutation Chromosome 2	1, 4, 6, 7, 3, *7, 7, 1, 3, 7, 6, 2, 2, 6,* 3, 8

Mutation operation is showed with the probability *P*m. Yet again, a random no. *R* is created in the range 0-1 and the individual undergoes mutation if and only if *R*<*P*m. Mutation operation will undergo as follows:

$$Pm=\begin{cases} K3 & f2>fav \\ K4 & f2<fav \end{cases}$$

Where f_2 shows the fitness value of the mutating chromosome and f_{av} is the avg. fitness value of the population.

Different values of K_3 and K_4 can be changed to find out a near optimal value of the Makespan. Guidelines by DeJong and Grefenstette for the selection of mutation rate are as follows:

Mutation rate = 0.001 (For large population size; 100)

Mutation rate = 0.01 (For small population size; 30)

Based on this guideline Mutation rates K_3 and K_4 were selected as follows:

Crossover Rate K_3 = 0.2

Crossover Rate K_4 = 0.1

Selection

The selection algorithm by the use of Roulette Wheel Selection is as follows:

(1) Fitness of all the population members is summed up. We can call it f_s .

(2) A random number 'R' is chosen between 0 and f_s .

(3) Fitness of the population members is added together one by one until the value of random number 'R' is reached in between the two values of the f_s . The last individual added is the selected individual and copy is added to the New Population which keeps on increasing with the new generations.

(4) This selection mechanism is continued until N individuals have been selected.

Gantt chart of Schedule – Example Problem:

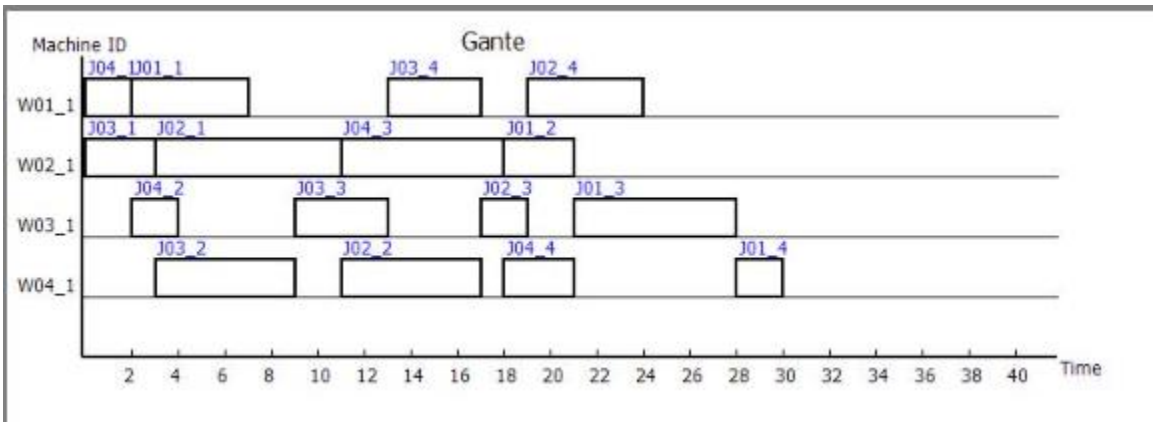


Figure 9: Shows the Gantt chart

2.5. Introduction to FJSS Software

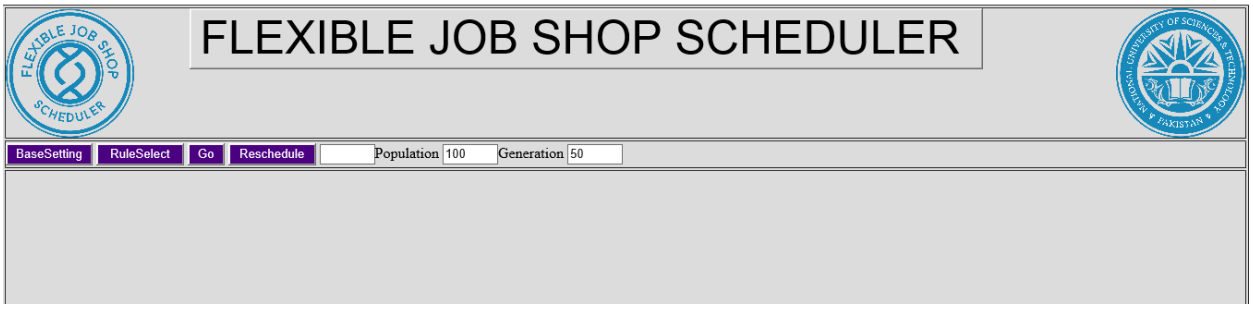
The first and the crucial information required to solve a scheduling problem of shop floor is the number of machines available in each work centers along with the number of jobs.

A flow chart is shown for better understanding of how input can be given,

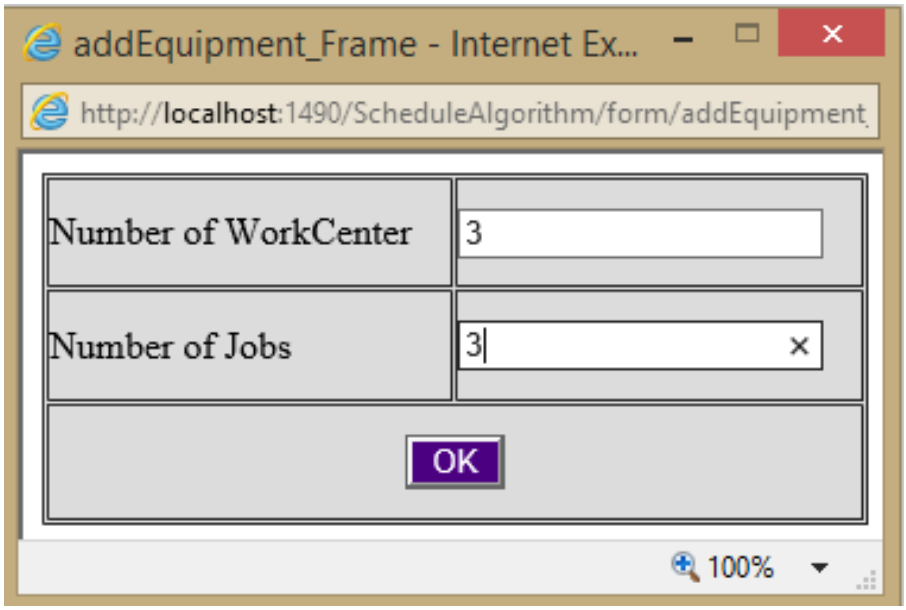
Figure 10: Flow Chart of Base Setting

2.5.1. Base Settings

When the program is run initially its generates a form that set up to take the input for the no. of Work Centers and the no. of Jobs. In addEquipment_Frame window this information is displayed with the start of Base Setting.

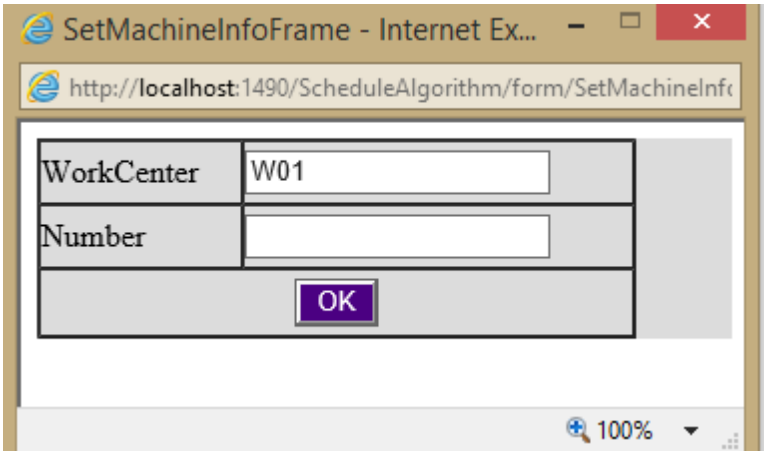


For work centers and number of jobs only numeric input can be done, it does not take the text values. Furthermore, the boxes of no of jobs and work centers cannot be left blank. The minimum value for the work center and number of jobs is "1".

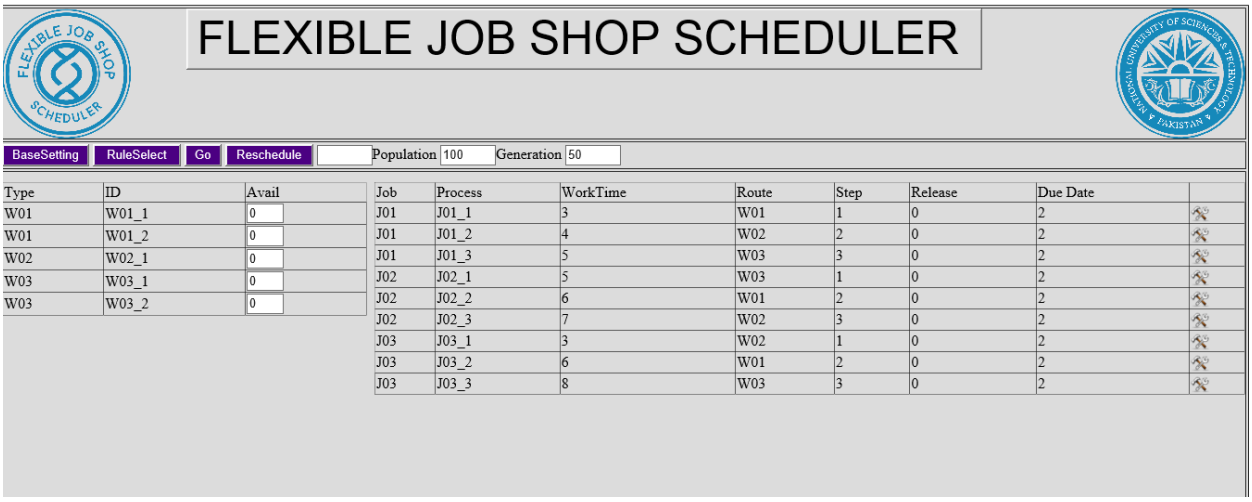


2.5.2. Processing and Machine Information

After the completion of no. of work center and no. of jobs, input for Number of identical machines in each of the work center is done. Processing time for every process of the job is done in SetJobInfoFrame.



No. of Work center and the no. of machines information in each work center is saved in the work center table (centerTable) with the work centers as “Type” column, machines within the work center under ”ID” column. Another column for availability “Avail” of machines is also included which can be used at later stage to input “Initial Setup Time”

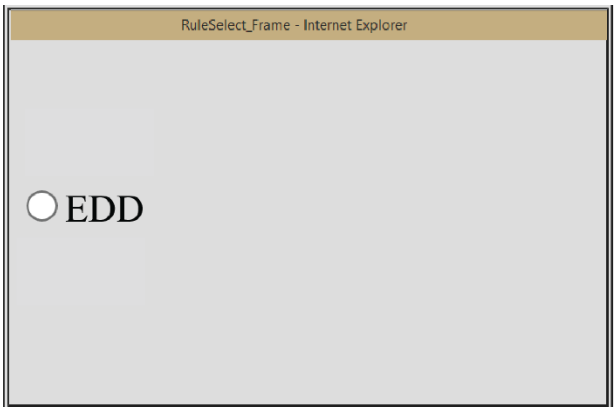


Type	ID	Avail
W01	W01_1	0
W01	W01_2	0
W02	W02_1	0
W03	W03_1	0
W03	W03_2	0

Job	Process	WorkTime	Route	Step	Release	Due Date
J01	J01_1	3	W01	1	0	2
J01	J01_2	4	W02	2	0	2
J01	J01_3	5	W03	3	0	2
J02	J02_1	5	W03	1	0	2
J02	J02_2	6	W01	2	0	2
J02	J02_3	7	W02	3	0	2
J03	J03_1	3	W02	1	0	2
J03	J03_2	6	W01	2	0	2
J03	J03_3	8	W03	3	0	2

Added table is also set up as “jobTable” to save the job information such as processing time, route etc. A sample code is added below which is used to create new “centerTable” and “jobTable”.

```
Private void BaseSetting_ServerClick(object sender, System.EventArgs e)

{

Session.RemoveAll();

//create a workcenter table in order to store and set workcenter information

System.Data.DataTablecenterTable=newSystem.Data.DataTable();

centerTable.Columns.Add(new DataColumn(“Type”));

centerTable.Columns.Add(newDataColumn(“ID”));
centerTable.Columns.Add(“Avail”,typeof(double));

Session[“centertable”]=centerTable;


//create a Job table in order to store and set job information

System.Data.DataTable jobTable=new System.Data.DataTable();

jobTable.Columns.Add(new DataColumn(“Job”));

jobTable.Columns.Add(newDataColumn(“Process”));

jobTable.Columns.Add(“WorkTime”,typeof(double));

jobTable.Columns.Add(newDataColumn(“Route”));
jobTable.Columns.Add(“Step”,typeof(int));

jobTable.Columns.Add(“Release”,typeof(double));
jobTable.Columns.Add(“DueDate”,typeof(int));

Session[“jobtable”]=jobTable
```



After the scheduling problem is established then the selection of rules is done in the RuleSelect_Frame to generate schedule in combination with the Genetic Algorithm.

2.5.3. Flow Chart of the FJSS Software

A complete flow chart of the scheduling software is given starting from the BaseSetting to the generation of schedule of GANTT CHART. This flow chart provides stage by stage information about the generation.

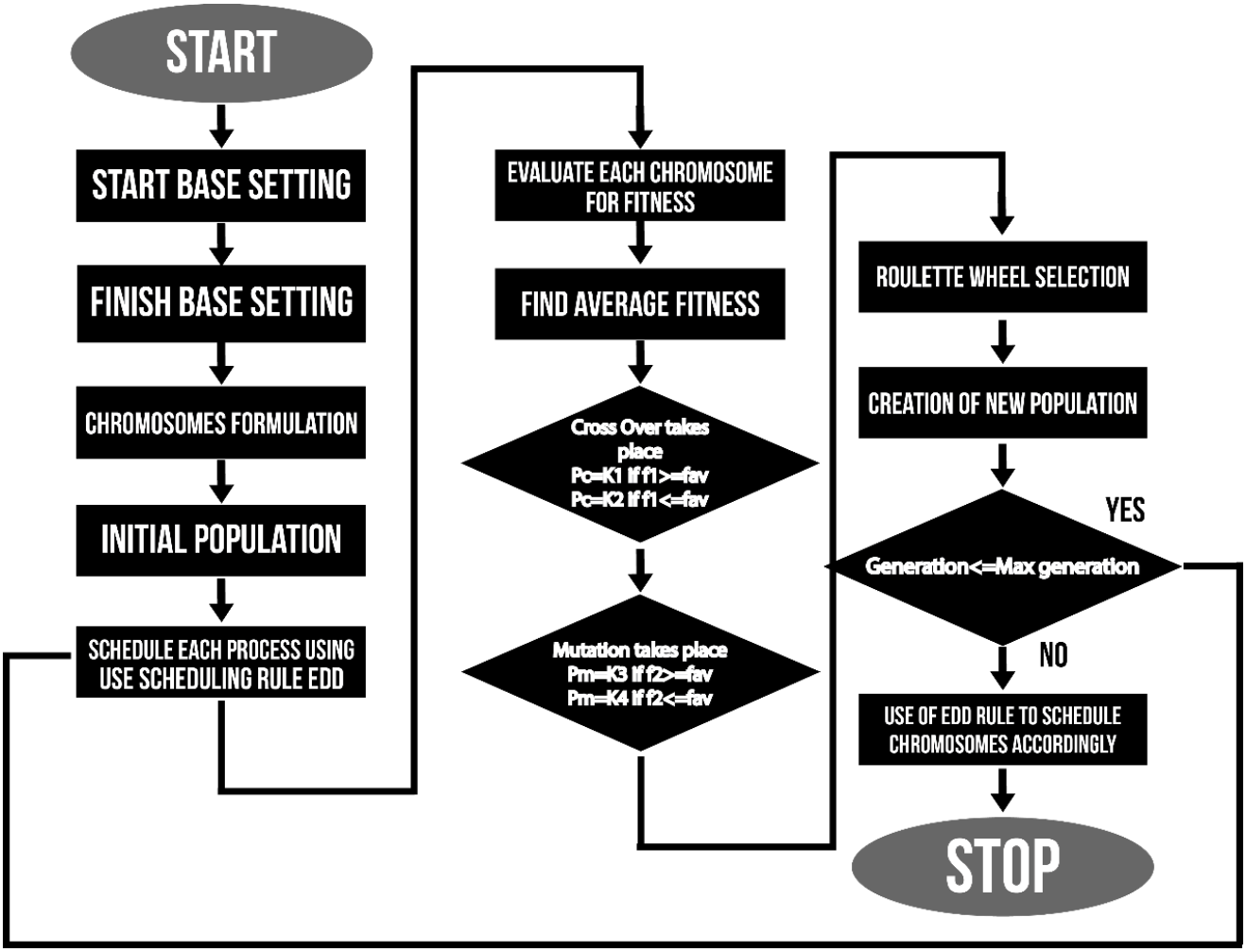


Figure 11: Flow Chart Flexible Job Scheduler Process

2.5.4. Initial Population

Below flow chart indicates the generation of initial population which is conferring to the population size chosen. Information for each gene (which represents the machine number) of the population is taken from the tables of JobTable and CenterTable stated above.

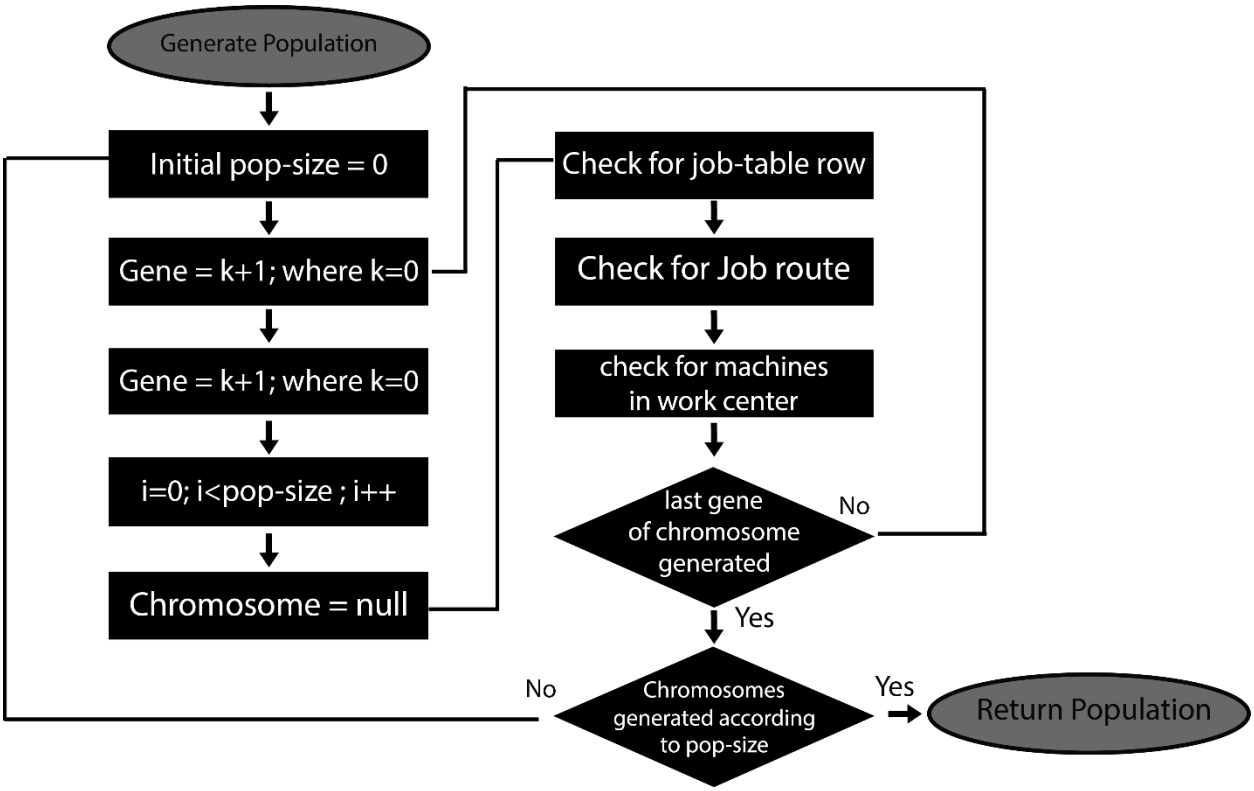


Figure 12: Flow Chart Flexible Job Scheduler Process

2.5.5. Fitness

Fitness of each chromosome and then in turn population fitness is calculated along with Average and Maximum fitness as shown in figures below, whereas the tie on each machine is broken according to the scheduling rule EDD.

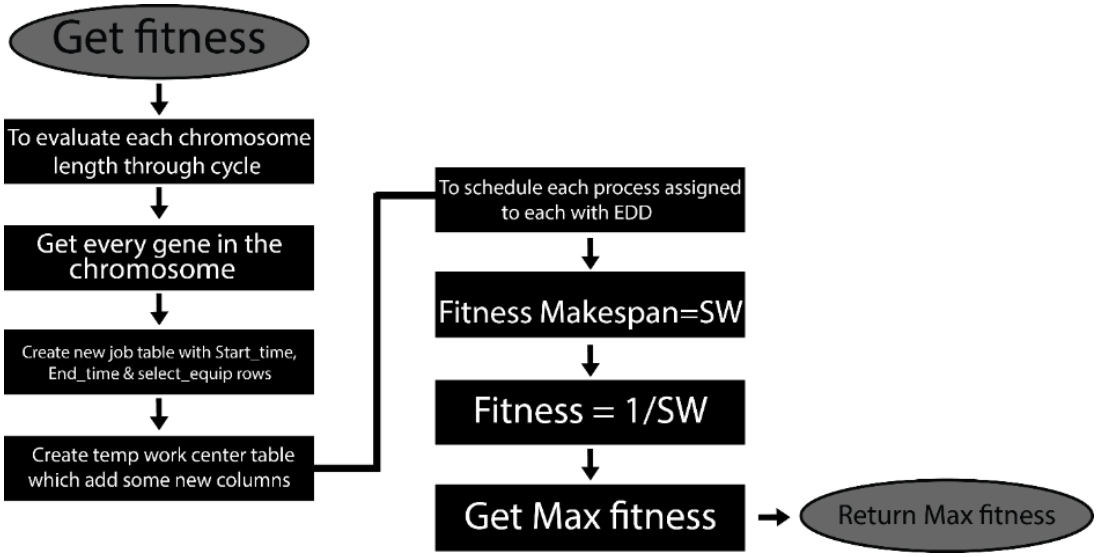


Figure 13: Flow Chart of Fitness

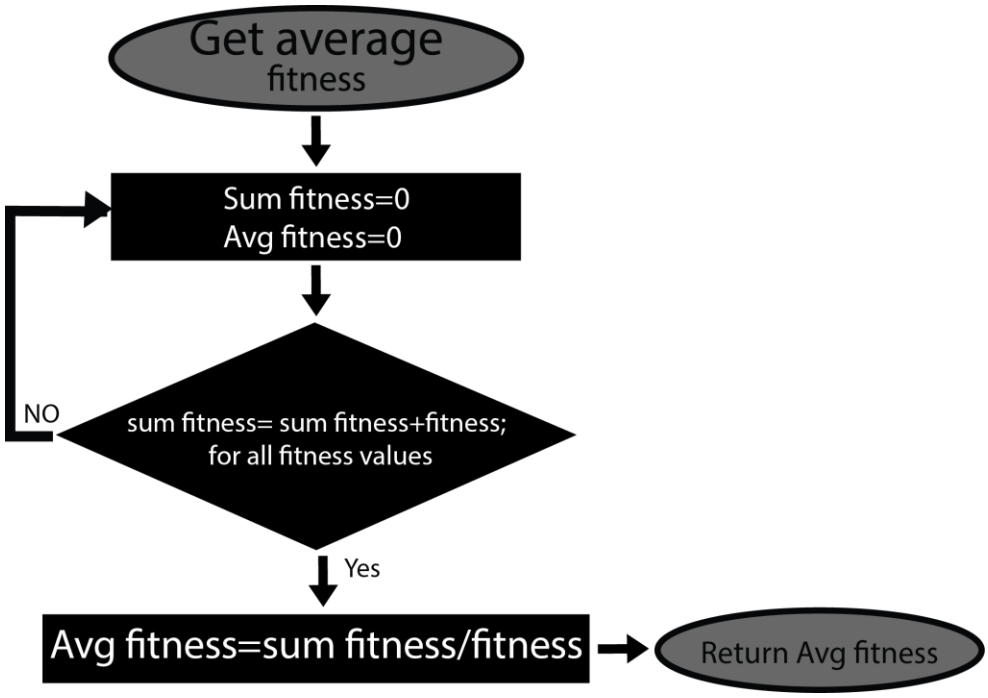


Figure 14: Flow Chart of Average Fitness

2.5.6. Crossover of Chromosome

Two-point crossover method is selected so that that two points are selected randomly. Crossover probability is selected according to the criteria mentioned below. Maximum fitness of the two chosen chromosome is compared with the Avg. Fitness of the population and if the Maximum Fitness is superior or equal to the Average Fitness then first Crossover probability is selected. If Maximum fitness of the two chosen chromosome is less than the average fitness then the second crossover probability is chosen. Maximum fitness is the chosen highest of the two parents. Random no. R is produced in the range 0-1 and the individual undergo crossover if and only if R<Pc otherwise no crossover probability is chosen. Figure shows the crossover of chromosome, whereas code is given below:

```
#region Cross-Over of Chromosome

public ArrayList CrossOver(Random RanI,Random RanD,ArrayList Fitness,ArrayList Population)

{

    ArrayList Cro_Chrom=new ArrayList();

    ArrayList Temp_Popu=new ArrayList(Population);

    ArrayList Temp_Fit=new ArrayList(Fitness);

    double AvgFitness=Get_AvgFitness(Fitness);

    double MaxFitness=Get_MaxFitness(Fitness);

    do

    {

        double mf=0;
```

```
int Seed=RanI.Next(1,Temp_Popu.Count);


mf=System.Math.Max((double)Temp_Fit[0],(double)Temp_Fit[Seed]);
string[]Parent1=Temp_Popu[0].ToString().Split(Convert.ToChar(","));
string[]Parent2=Temp_Popu[Seed].ToString().Split(Convert.ToChar(","));
if(CrossOper(RanD,mf,AvgFitness,MaxFitness)) {

        int Pos_1=0;

        int Pos_2=0;

        Pos_1=RanI.Next(0,Parent1.Length);

        do

        {

                Pos_2=RanI.Next(0,Parent1.Length);

        }while(Pos_2==Pos_1);

        int Length=System.Math.Abs(Pos_2-Pos_1)+1;

        int Pos=System.Math.Min(Pos_1,Pos_2);

        string[] Cross_1=new String[Length];

        string[] Cross_2=new String[Length];

        for(int i=0;i<Length;i++)
```

```
    {
        Cross_1[i]=Parent1[i+Pos].ToString();
        Cross_2[i]=Parent2[i+Pos].ToString();
}
for(int j=0;j<Length;j++)
{
        Parent1[Pos+j]=Cross_2[j].ToString();
        Parent2[Pos+j]=Cross_1[j].ToString();
}
string str_Par1="";
string str_Par2="";
for(int k=0;k<Parent1.Length;k++)
{
        str_Par1=str_Par1+Parent1[k].ToString()+",";
        str_Par2=str_Par2+Parent2[k].ToString()+",";
```

```
                }

                str_Par1=str_Par1.Remove(str_Par1.Length-1,1);

                str_Par2=str_Par2.Remove(str_Par2.Length-1,1);

                Cro_Chrom.Add(str_Par1);

                Cro_Chrom.Add(str_Par2);

        }

        Temp_Popu.RemoveAt(0);

        Temp_Popu.RemoveAt(Seed-1);

        Temp_Fit.RemoveAt(0);

        Temp_Fit.RemoveAt(Seed-1);

}while(Temp_Popu.Count!=0);

return Cro_Chrom;

}

private bool CrossOper(Random RanD,double mf,double AvgFitness,double MaxFitness)

{
```

```
double Pc=0;

    double R=RanD.NextDouble();//----(0,1)

    if(mf>=AvgFitness)

    {

            Pc=0.9;

    }

    else

    { Pc=0.85;}
    if(R<Pc)
    {

            return true;

    }

    else

    {return false;}

    }

    #endregion
```

2.5.7. Mutation of Chromosome

Mutation gene within the chromosome is selected in such that a point is selected randomly. Mutation Probability is selected according to criteria mentioned below. Comparison of Maximum fitness of the mutating chromosome is compared with average fitness of population and if the maximum fitness is better or equal to the average fitness then the first mutation probability is selected. If maximum fitness is less than the average fitness then second crossover probability is chosen. Random no. R is created in the range 0-1 and the individual will undergo mutation if and only if R<Pc otherwise no mutation probability is chosen. Figure shows the mutation of chromosome, whereas code is given below:

```
#region Mutation of Chromosome


public ArrayList Mutation(Random RanI,Random RanD,ArrayList Fitness,ArrayList
Population,DataTable newJobTable,DataTable tempCenterTable) {


    double AvgFitness=Get_AvgFitness(Fitness);


    double MaxFitness=Get_MaxFitness(Fitness);


    ArrayList Mut_Chrom=new ArrayList();


    for(int i=0;i<Population.Count;i++)


    {


    if(Mut_Oper(RanD,Population[i].ToString(),(double)Fitness[i], AvgFitness, MaxFitness))


    {


    string[]GeneList=Population[i].ToString().Split(Convert.ToChar(","));


    int Count=0;
```

```
DataRow[] EquipQue;

string Select_Equip="";

int Mut_Pos=0;



Mut_Pos=RanI.Next(0,GeneList.Length);

string Route=newJobTable.Rows[Mut_Pos]["Route"].ToString();

EquipQue=tempCenterTable.Select("Type='"+Route+"'");

Select_Equip=tempCenterTable.Rows[Convert.ToInt32(GeneList[Mut_Pos])-

1]["ID"].ToString();

Count=EquipQue.Length;

string new_Equip="";

int Seed=RanI.Next(0,Count);

new_Equip=EquipQue[Seed]["ID"].ToString(); //Selection of new mutation gene

    for(int j=0;j<tempCenterTable.Rows.Count;j++)

    {
```

```
            if(tempCenterTable.Rows[j]["ID"].ToString()==new_Equip)

                {

                    GeneList[Mut_Pos]=Convert.ToString(j+1);

                    }

            }

                    string strMutation="";

                    for(int k=0;k<GeneList.Length;k++)

                    {

                    strMutation=strMutation+GeneList[k]+",";

                    }

                    if (strMutation.Length>1)

                    {

                        strMutation=strMutation.Substring(0,strMutation.Length-1);

                    }

                    Mut_Chrom.Add(strMutation);

        }
```

```
    }

            return Mut_Chrom;

}

private bool Mut_Oper(Random RanD,string Parent,double fm,double AvgFitness,double
MaxFitness)

        {

            double Pm=0;

            if(fm<AvgFitness)

            {Pm=0.1;}

            else

            {

                Pm=0.2;

            }

                double R=RanD.NextDouble();

                if(R>=Pm)
```

```
                {return false;}
            return true;

}

#endregion
```

2.5.8. Selection Process

All the chromosomes that are generated during the process of mutation and crossover process are added up and become the part of the total population in the selection process. Which results in increase population size and the selection process takes place. This selection process takes place with the summing up of ratio of population fitness, which is denoted with 'Sp' in the code. Random No. 'R' is created whose value may vary between '0' and 'Sp'. Thus, the last individual's value, which is added up to equal to the Random number 'R', is selected and copied to the New Population. Selection process is continued until N individuals have been chosen for the new population. Figure 4-19 gives a flow chart for the selection process by Roulette selection.

2.5.9. Initial Setup Time

Every machine requires some initial setup time to begin the process of the job, in this scheduler a special case of initial setup time is being used. Setup time is usually described as the time interval for cleaning interval changing tools, cleaning tools and operations etc. Setup time add complications to the real scheduling problems. There are two general categories of the setup time are sequence dependent setup times and sequence independent setup times. First type of times are general and complex while second type of times might be additional to the processing times.

In the suggested scheduler software setup times is only considered only at the beginning of the first process whereas the remaining setup times between jobs are considered either zero or included in the process times.

Provision to add initial setup times is provided in the main Form under the column of 'Avail. This time represents the initial setup times for different machines in the work centers. The initial setup times will remain constant and are included in the scheduling and rescheduling process. The sub-routine is provided under the class of Heuristics Rule to make Initial Set Up Times part of the schedule and reschedule.

2.5.10. Summary

This chapter gives a brief insight of the basic concepts and the working of the genetic algorithm. A simple genetic algorithm along with its basic components is presented. An example of manual calculation is presented from David E Goldberg book, which is a very adequate example in understanding the concepts of genetic algorithm. FJSP scheduling software is enlightened with the help of the screenshots and flow charts. Genetic algorithm along with Earliest Due Date (Scheduling rule) is used to develop the Gantt chart with the objective of Makespan Time. Functionality of this software will be checked by solving some problems in next chapter.

CHAPTER 3

EXPERIMENTATION AND RESULTS

3. EXPERIMENTATION

This chapter presents the details of the experimentations conducted on the flexible job shop scheduler. Software established presently combines GA along with the rule EDD (Earliest Due Date) with an possibility to add more rules as necessary. As the focus of our research is on Flexible Job Shop, so the Job Shop scheduling problem was converted to the Flexible Job Shop scheduling problem based on two simple criteria which was proposed by Barnes and Chambers [23]:

The first criterion is the total processing time required by a machine. This can be easily calculated from the problem data and is interesting in that a planner might spontaneously expect improvement from the replication of a machine for which it is known demand will be high. The second criterion used is the cardinality of critical operations on a machine, based on the best solution obtained to the classical problem. Thus based on the above mentioned criterion replication policies are developed to convert Classical Job Shop problems to the Flexible Job Shop problems. Replication policies for the machines are listed below in the table. Processing times for operations on replicated machines are assumed to be identical to the original.

Instances	**Description**
p1,p1,p1	The machine requiring the greatest processing time is replicated three times.
P1,P2	The machines requiring the greatest and second- greatest processing times are replicated once each.
p1,p2,p3	The machines requiring the greatest, second- greatest and third greatest processing times are replicated once each.
p1,p2,p3,p4	**The machines requiring the greatest, second- greatest, third greatest and fourth greatest processing times are replicated once each**
c1	The machine with the greatest number of critical operations is replicated once
c1,c2	The machines with the greatest and second greatest number of critical operations are replicated once each
C1,c2,c3	The machines with the greatest, second greatest and third greatest number of critical operations are replicated once each

3.1. Test Problems 1 – FT06 Benchmark

The FT06 benchmark problem was developed by Fisher and Thomson (1963)[24]. The problem contains a set of 6 jobs to be processed on 6 different machines, where every job has its own machining sequence. The due dates for this problem are taken from the work of Ponnambalam, et al. (2001)[25]. Table shows the processing times (min) and due dates (min) for the FT06 problem. Instance 4 from Barnes and Chambers techniques is selected for the conversion of JSP to FJSSP

Table 10: Shows the processing time for FT06

Jobs	M1	M2	M3	M4	M5	M6
Job 1	1	3	6	7	3	6
Job 2	8	5	10	10	10	4
Job 3	5	4	8	9	1	7
Job 4	5	5	5	3	8	9
Job 5	9	3	5	4	3	1
Job 6	3	3	9	10	4	1

Table 11: Shows the Machine Sequence for FT06

Jobs	M1	M2	M3	M4	M5	M6
Job 1	3	1	2	4	6	5
Job 2	2	3	5	6	1	4
Job 3	3	4	6	1	2	5
Job 4	2	1	3	4	5	6
Job 5	3	2	5	6	1	4
Job 6	2	4	6	1	5	3

Table 12: Shows the Due Date for FT06

Jobs	Job 1	Job 2	Job 3	Job 4	Job 5	Job 6
Due Date	52	94	68	70	25	45

3.2. RESULTS

In this section, the results for FT06 problem tested are presented. The trends observed for the test problem are explained separately under the respective sections.

Table 13: Shows results generated for FT06 problem for JSS

Job	Process	WorkTime	Select_Equip	Start_Time	End_Time	Due Date	Tardiness
J01	J01_1	1	W03_1	9	10	52	
J01	J01_2	3	W01_1	13	16	52	
J01	J01_3	6	W02_1	19	25	52	
J01	J01_4	7	W04_1	25	32	52	
J01	J01_5	3	W06_1	32	35	52	
J01	J01_6	6	W05_1	36	42	52	0
J02	J02_1	8	W02_1	8	16	94	
J02	J02_2	5	W03_1	20	25	94	
J02	J02_3	10	W05_1	49	59	94	

J02	J02_4	10	W06_1	59	69	94	
J02	J02_5	10	W01_1	69	79	94	
J02	J02_6	4	W04_1	79	83	94	0
J03	J03_1	5	W03_1	10	15	68	
J03	J03_2	4	W04_1	15	19	68	
J03	J03_3	8	W06_1	19	27	68	
J03	J03_4	9	W01_1	27	36	68	
J03	J03_5	1	W02_1	36	37	68	
J03	J03_6	7	W05_1	42	49	68	0
J04	J04_1	5	W02_1	3	8	70	
J04	J04_2	5	W01_1	8	13	70	
J04	J04_3	5	W03_1	15	20	70	
J04	J04_4	3	W04_1	20	23	70	
J04	J04_5	8	W05_1	24	32	70	
J04	J04_6	9	W06_1	35	44	70	0
J05	J05_1	9	W03_1	0	9	25	
J05	J05_2	3	W02_1	16	19	25	
J05	J05_3	5	W05_1	19	24	25	
J05	J05_4	4	W06_1	27	31	25	
J05	J05_5	3	W01_1	36	39	25	
J05	J05_6	1	W04_1	39	40	25	**15**
J06	J06_1	3	W02_1	0	3	45	
J06	J06_2	3	W04_1	3	6	45	
J06	J06_3	9	W06_1	6	15	45	
J06	J06_4	10	W01_1	16	26	45	
J06	J06_5	4	W05_1	32	36	45	
J06	J06_6	1	W03_1	36	37	45	0

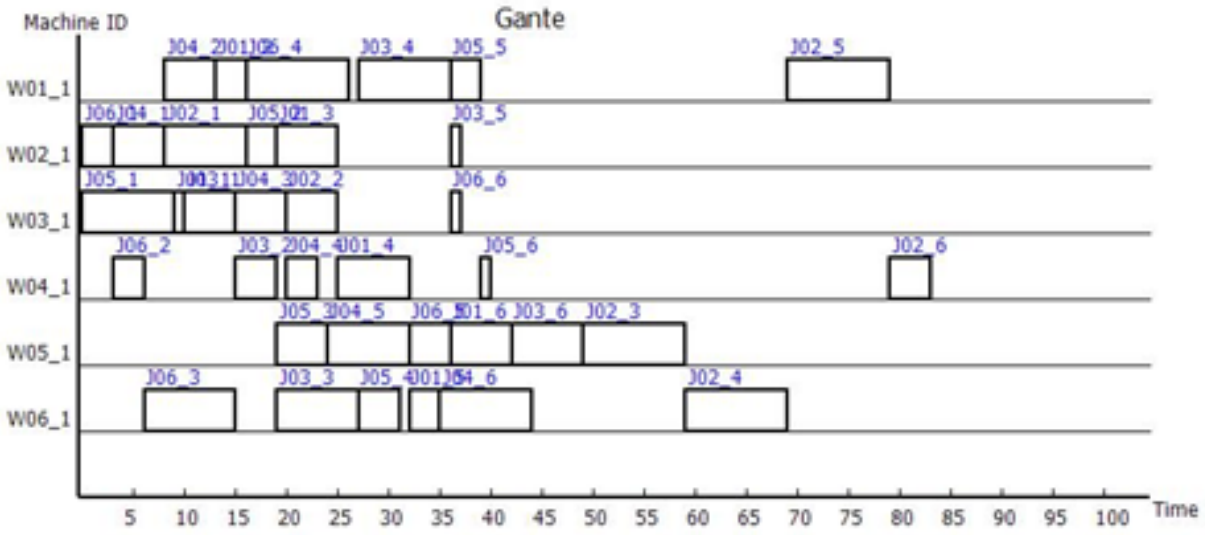


Figure 15: Schedule Gantt chart for FT06 Problem for JSS

Table 14: Shows results generated for FT06 problem for FJSS

Job	Process	WorkTime	Select_Equip	Start_Time	End_Time	Due Date	Tardiness
J01	J01_1	1	W03_1	9	10	52	
J01	J01_2	3	W01_2	10	13	52	
J01	J01_3	6	W02_1	14	20	52	
J01	J01_4	7	W04_1	20	27	52	
J01	J01_5	3	W06_2	27	30	52	
J01	J01_6	6	W05_1	35	41	52	0
J02	J02_1	8	W02_1	3	11	94	
J02	J02_2	5	W03_1	20	25	94	
J02	J02_3	10	W05_1	25	35	94	
J02	J02_4	10	W06_2	35	45	94	
J02	J02_5	10	W01_1	45	55	94	
J02	J02_6	4	W04_1	55	59	94	0
J03	J03_1	5	W03_1	10	15	68	
J03	J03_2	4	W04_1	15	19	68	
J03	J03_3	8	W06_2	19	27	68	
J03	J03_4	9	W01_1	27	36	68	
J03	J03_5	1	W02_1	36	37	68	
J03	J03_6	7	W05_1	41	48	68	0
J04	J04_1	5	W02_2	0	5	70	
J04	J04_2	5	W01_2	5	10	70	
J04	J04_3	5	W03_1	15	20	70	

J04	J04_4	3	W04_1	27	30	70	
J04	J04_5	8	W05_2	30	38	70	
J04	J04_6	9	W06_1	38	47	70	0
J05	J05_1	9	W03_1	0	9	25	
J05	J05_2	3	W02_1	11	14	25	
J05	J05_3	5	W05_1	14	19	25	
J05	J05_4	4	W06_1	19	23	25	
J05	J05_5	3	W01_2	25	28	25	
J05	J05_6	1	W04_1	30	31	25	6
J06	J06_1	3	W02_1	0	3	45	
J06	J06_2	3	W04_1	3	6	45	
J06	J06_3	9	W06_2	6	15	45	
J06	J06_4	10	W01_2	15	25	45	
J06	J06_5	4	W05_2	25	29	45	
J06	J06_6	1	W03_1	29	30	45	0

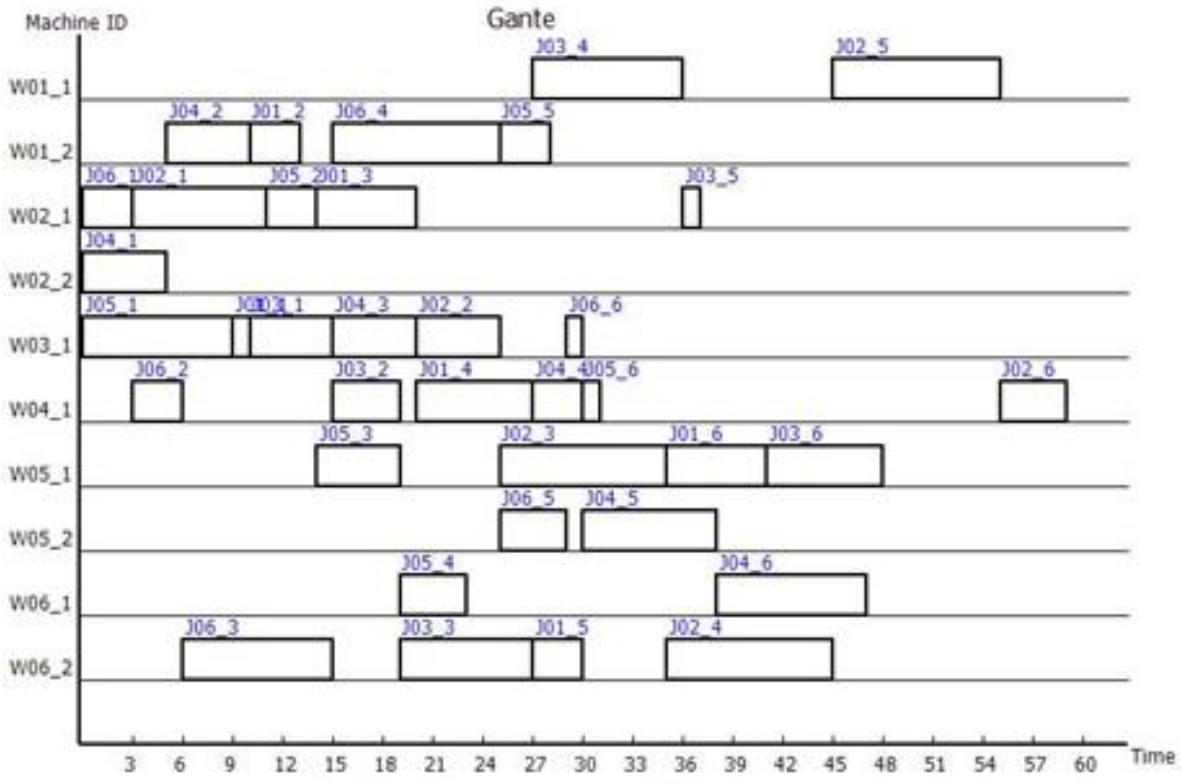


Figure 16: Schedule Gantt chart for FT06 Problem for FJSS

Table 15: Shows GA + EDD rule search results (makespan) for FT06 problem

Test Problem	No of Machines	No of Jobs	Makespan (min) (Results by Ponnambalam) *JSSP Multiobjective GA	Makespan Time (our Research) GA + EDD JSSP	% Gap btwn the previous and our research JSSP	Makespan Time (our Research) GA + EDD FJSSP	% Gap btwn the previous JSSP and our research FJSSP
FT06	6	6	96	83	7.26%	59	23.87

It is evident from Table 15 results that benchmark problem (FT06) solved by multiobjective genetic algorithm with due date technique for job shop scheduling problem by Ponnambalam has a makespan time of 96 min while it was attained as 83 min with proposed hybrid algorithm. The improvement in the makespan came out to be 7.26%.

Further it is evident that if the problem is converted to FJSSP using Barnes and Cambers techniques, the make-span times improves further to a lower value of 59 min with further improvement of 23.87% with proposed hybrid algorithm.

Table 16: Shows GA + EDD rule search results (total tardiness) for FT06 problem

Test Problem	No of Machines	No of Jobs	Total Tardiness (min) (Results by Ponnambalam) *JSSP Multiobjective GA	Total Tardiness (our Research) GA + EDD JSSP	% Gap btwn the previous and our research JSSP	Makespan Time (our Research) GA + EDD FJSSP	% Gap btwn the previous JSSP and our research FJSSP
FT06	6	6	32	15	36.17%	6	68.42%

Similarly it is evident from table 16 that benchmark problem (FT06) solved by multiobjective genetic algorithm with due date technique for job shop scheduling problem by Ponnambalam has

a total tardiness of 32 min while it was attained as 15 min with proposed hybrid algorithm. The improvement in the makespan came out to be 36.17%.

Further it is evident that if the problem is converted to FJSSP using Barnes and Cambers technique the make-span times improves further to a lower value of 6 min with further improvement of 68.42% with proposed hybrid algorithm.

3.3. FUTURE DIRECTIONS

This thesis provides a great possibilities for the new researchers to use their energies for the advancement of the algorithm and the software. One of the prospective future directions is the integration of the genetic algorithm with any other evolutionary algorithm ie Ant Colony Optimization. Second direction is to add different scheduling rules to the program. Third direction is to use different objective function such as mean tardiness and mean earliness and then calculate cost of job earliness and job tardiness.

CHAPTER 4

REFERNCES

[1] Manufacturing Systems Engineering: A Unified Approach to Manufacturing ... By Katsundo Hitomi

[2] V.C.S. Wiers (1997) Human Computer Interaction in Production Scheduling: Analysis and Design of Decision Support Systems in Production Scheduling Tasks, Ph.D Thesis, Eindhoven University of Technology, Eindhoven, the Netherlands.

[3] Noor , J.A.E (2007) Electrical Impedance Tomography at low frequencies (Phd Thesis) . Kensington, Australia, University of New South Wales Noordegraaf

[4] R.W. Conway (1965a) "Priority Dispatching and Work-In-Process Inventory in a Job Shop", Journal of Industrial Engineering, Vol. 16, pp. 123–130

[5] V. Jain and I.E. Grossmann (2001) "Algorithms for Hybrid MILP/CP Models for a Class of Optimization Problems", Informs Journal on Computing, Vol. 13, pp. 258–276.

[6] Brucker, P. and R. Schlie, Job-shop scheduling with multi-purpose machines. Computing, 1990. 45(4): p. 369-375.

[7] Barnes, J.W.C., J. B, Flexible Job Shop Scheduling by Tabu Search, T.R.S. Graduate Program in Operations Research and Industrial Engineering, ORP96-09, Editor. 1996, University of Texas at Austin

[8] Najid, N.M., S. Dauzere-Peres, and A. Zaidat, A modified simulated annealing method for flexible job shop scheduling problem, in IEEE International Conference on Systems, Man and Cybernetics, A.E. Kame, K. Mellouli, and P. Borne, Editors. 2002: Tunisia. p. 6 pp

[9] Kacem, I.H., Slim; Borne, Pierre, Approach by localization and multiobjective evolutionary optimization for flexible job-shop scheduling problems. Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on, 2002. 32(1): p. 1-13.

[10] Chan, F.T.S.W., T. C.; Chan, L. Y., Flexible job-shop scheduling problem under resource constraints. International Journal of Production Research, 2006. 44(11): p. 2071-2089.

[11] Darwin, C., On the origin of species by means of natural selection. 1859. Ed. Joseph Carroll. Toronto: Broadview, 2003.

[12] Holland, J.H., Adaptation in natural and artificial systems. 1975, Ann Arbor, MI: University of Michigan Press.

[13] Goldberg, D.E., Genetic algorithms in search, optimization and machine learning. 1989: Addison-Wesley Longman Publishing Co., Inc.

[14] Yamada T. Studies on Metaheuristics for Jobshop and Flowshop scheduling problems [D] Kyoto: Kyoto University, Japan, 2003

[15] Ponnambalam S G, Aravindan P, Sreenivasa R P. Comparative evaluation of genetic algorithms for job-shop scheduling [J]. Production Planning & Control. 2001, 12(6): 560574

[16] Deepu Philip, Scheduling reentrant flexible job shops with sequence dependent set up times [D]. Montana: Montana State University, 2005

[17] FOGARTY, M. J., SISSENWINE, M. P. and E. B. COHEN 1991 — Recruitment variability and the dynamics of exploited marine populations. Trends Ecol. Evol. 6(8): 241–246

[18] Operations management: concepts, methods, and strategies Vonderembse, Mark A., 1948- ; White, Gregory P., 1948-

[19] Intelligent Scheduling with machine learning capabilities; The introduction of scheduling knowledge ,'HE Transactions 24, 156-168. Singh, M., and T.E. Morton (1989). "Non-Premptive Scheduling to Minimize the Maximum Latness.

[20] Application of Genetic Algorithm and rules in the Scheduling of Flexible Job Shop by Shahid Ikramullah and Sun Hou- Fang

[21] Dejong K A, Spears W M. Ana analysis of the interacting roles of population size and crossover in genetic algorithms [C]// Proceedings of first workshop parallel problem solving from nature, Springer-Verlag, Berlin. 1990: 38-47.

[22] Grefenstette J J. Optimization of control parameters for genetic algorithms [J]. IEEE Transactions on System, Man and Cybernetics. 1986, 6(1): 122-128

[23] Barnes, J.W.C., J. B, Flexible Job Shop Scheduling by Tabu Search, T.R.S. Graduate Program in Operations Research and Industrial Engineering, ORP96-09, Editor. 1996, University of Texas at Austin.

[24] Fisher, H. and G.L. Thompson, Probabilistic learning combinations of local job-shop scheduling rules. Industrial scheduling, 1963. 3(2): p. 225-251

[25] Ponnambalam, S. G., Ramkumar, V., and Jawahar, N., (2001), "A multiobjective genetic algorithm for job shop scheduling", Journal of Production Planning and Control, Vol. 12, No. 8, pp. 764-774.