Optimization of Flexible Flow Shop Scheduling Using Genetic Algorithm

A Project Report

Submitted by

B.P.M. Harikrishnaraj	-	71202114015
K. Harish Kumar	-	71202114016
M. Manoj Prabakar	-	71202114020

in partial fulfillment for the award of the degree
of

Bachelor of Engineering
in
Mechanical Engineering

DEPARTMENT OF MECHANICAL ENGINEERING

KUMARAGURU COLLEGE OF TECHNOLOGY

COIMBATORE – 641 006

ANNA UNIVERSITY :: CHENNAI 600 025

APRIL– 2006

ANNA UNIVERSITY :: CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report entitled "**Optimization of Flexible Flow Shop Scheduling Using Genetic Algorithm**" is the bonafide work of

Mr.B.P.M.Harikrishnaraj	-	Register No. 71202114015
Mr.K.Harish Kumar	-	Register No. 71202114016
Mr.M.Manoj Prabhakar	-	Register No. 71202114020

Who carried out the project work under my supervision.

Signature of the Head of the Department

Dr.T.P.Mani

HEAD OF THE DEPARTMENT

Signature of the supervisor

Mr.K.Raja M.E

SUPERVISOR

Internal Examiner 28/04/04

Dr. T.P. Mani

B.E., M.E., Ph.D., DML, MIE, MISTE, MISC

Dean & HoD / Dept. of Mech. Engg.

Kumaraguru College of Technology

Coimbatore - 641 006

External Examiner 28/4/04

DEPARTMENT OF MECHANICAL ENGINEERING

KUMARAGURU COLLEGE OF TECHNOLOGY

COIMBATORE 641 006

QPT/SEC/PRO/2006

Dated: 21.04.2006

TO WHOMSOEVER IT MAY CONCERN

This is to certify that the following students

Mr.B.P.M. Harikrishnaraj - Reg. No. 71202114015

Mr. K.Harish Kumar - Reg. No. 71202114016

Mr.M.Manoj Prabakar - Reg. No. 71202114020

of Final year B.E (Mechanical Engineering), Kumaruguru College of Technology, Coimbatore have done a project work on **"OPTIMIZATION OF FLEXIBLE FLOW SHOP SCHEDULING USING GENETIC ALGORITHM"** in our organization during the period January 2006 to April 2006.

During the course of the project work their conduct and decorum was good. They have successfully completed the project.

For **Q PLUS TECHNOLOGIES**

Authorized Signatory

#26 Mahu Nagar,Opp Ganapathy Telephone Exchange,Ganapathy ,Coimbatore-641 006

0422-2530092 0422-2521592

ABSTRACT

Manufacturing industries have become the most important contributors to prosperity. Industries must be able to adjust quickly to the enduring market conditions and has to maximize their utilization of available resources. Scheduling can improve on time delivery, reduce inventory cost, cut lead time and improve the utilization of available resources. Because of the combinatorial nature of scheduling problems, it is often hard to obtain optimal schedules especially with a limited period of computational time. The problem discussed in this report involves flexible flow shop scheduling.

The objective is to find a schedule which assigns jobs to the machines in such a way that the makespan, earliness and tardiness are minimized. Minimizing the makespan, earliness and tardiness is important since it leads to the best use of system resources. Recent trends in scheduling research are aimed towards developing models that are more relevant to the practical situation. Genetic algorithm is good at handling large amount of data and works well in search space. Genetic algorithm depicts natural gene reproduction processes. Genetic algorithm is a simple, fast and convenient tool for solving a broad class of very hard combinatorial optimization problems.

A typical industrial problem of scheduling jobs through parallel machines is taken up for in-depth analysis. The process times of the jobs are deterministic. The combined objective is to minimize makespan, earliness and tardiness. Visual Basic programming has been done for the entire Genetic Algorithm procedure with available data and the outcome of the Genetic Algorithm technique has been compared and analyzed graphically. It has been found that the proposed genetic algorithm technique gives optimized results.

ACKNOWLEDGEMENT

We extend our profound gratitude to our amicable guide **Mr.K.Raja** Senior Lecturer. His valuable and uncountable contribution was meticulous. His relevant suggestions and appropriate corrections were helpful in the completion of this project successfully.

We place on record our sincere gratitude to our Dean and H.O.D Mechanical Engineering **Dr.T.P.Mani** for his progressive outlook and valuable suggestions.

Not out of formality, but out of sheer gratitude we would like to express our thank fullness to Principal **Dr.K.K.Padmanaban** for providing us with all facilities to finish the project.

We are immensely grateful to all the departments of **"Q Plus Technologies"** for providing us with all facilities to finish the project.

We also wish to express our special appreciation to all our friends, faculty and well-wishers for their timely help and encouragement.

Finally, we would like to thank our family members for their support in all our endeavors.

ORGANISATION PROFILE OF Q PLUS TECHNOLOGIES

Q Plus Technologies was started in 2004. The promoters are technocrats and specialists in CNC field with more than 20 years of experience from Lakshmi Machine Works Ltd and Craftsman Automation (P) Ltd. With the vast experience from these finest engineering companies of Coimbatore, further aided with the Q Plus infrastructure, customer satisfaction is ensured at all levels of the business. Q Plus aims to add more value to customer's products with high quality at low cost.

PRODUCTS

Q Plus is manufacturing all precision components to meet customer requirements. The various parts which are used in CNC machine tools, aerospace, automobiles, textiles, pumps and other engineering industries are machined. The company is equipped with a good design department for designing special jigs and fixtures.

EXPORTS

Various machined components are being exported to U.S.A, France and Italy.About 30% of the total turn over is being incurred from its exports.

CUSTOMERS

English Tools and Castings (P) Ltd, Souriau – India (Aerospace components), Marks Engineering Works, Pricol, Numinous Impex India (P) Ltd , CPC (P) Ltd, GTN Exports, Anugraha Valves, Precicraft Vacuum Pumps (P) Ltd and Auto Die Cast (P) Ltd.

ACHIEVEMENTS

- Rejection level of the machined components has been reduced to less than 2%.
- Rapid increase in the international customers since its inception.

CONTENTS

Title		PageNo.
Certificate		iii
Abstract		iv
Acknowledgement		v
Company's profile		vi
Contents		vii
List of Tables		xi
List of Figures / Graphs / Photos		xii
List of Abbreviations		xiii
CHAPTER 1 INTRODUCTION		**1**
1.1	Scope and purpose	1
1.2	Sequence of the Report	2
1.3	Limitations of the Project	2
1.4	Literature survey	2
1.4.1	Other Related Studies	4
CHAPTER 2 SCHEDULING		**6**
2.1	Introduction	6
2.2	Single Machine Scheduling	7
2.2.1	Processing Time	7
2.2.2	Ready Time	7
2.2.3	Flow Time	8
2.2.4	Lateness	8
2.2.5	Tardiness	8
2.3	Parallel Machine Scheduling	8
2.4	Job Shop Scheduling	9
2.5	Flow Shop Scheduling	10

CHAPTER 3 OBJECTIVES AND PROBLEM IDENTIFICATION		**11**
3.1	Objectives	11
3.2	Problem Identification	11
3.3	Problem Definition	12
CHAPTER 4 PROBLEM SOLVING		**14**
4.1	Problem Solving Procedure	14
4.2	Problem Formulation	15
4.2.1	Earliness and Tardiness	15
4.2.2	Makespan	15
4.2.3	Combined Objective Function (C.O.F)	15
4.3	Proposed Methodology	15
4.4	Procedure for Applying GA	16
4.4.1	Input for Algorithm	16
4.4.2	GA Representation	16
CHAPTER 5 METHODOLOGY		**18**
5.1	Genetic Algorithm	18
5.1.1	History of GA	19
5.1.2	Evolution	19
5.1.3	Basic Concept of GA	20
5.1.4	Difference and Similarities between GA and other Traditional Methods	21
5.2	Structure of GA	22
5.3	Basic Terms of GA	23
5.3.1	Chromosome	23
5.3.2	Candidate Solution	23
5.3.3	Fitness	23
5.3.4	Fitness Function	24
5.4	Control Parameters	24
5.5	Components of GA	24
5.5.1	Population Size	24
5.5.2	Cross over Probability	25
5.5.3	Mutation Probability	25
5.5.4	Encoding	25

5.5.4.1	Permutation Encoding	26
5.6	Genetic Operations	27
5.6.1	Reproduction	27
5.6.1.1	Rank Selection	27
5.6.2	Crossover	27
5.6.2.1	One point Crossover	28
5.6.2.2	Two point Crossover	28
5.6.2.3	Multi point Crossover	28
5.6.3	Mutation	28
5.7	Operation of GA	29
5.7.1	Initialization	29
5.7.2	Selection	29
5.7.3	Reproduction	30
5.7.4	Crossover	30
5.7.5	Mutation	31
5.8	Advantages of GA	31
5.8.1	Population based searches	31
5.8.2	Major advantages	32
5.9	GA applications	32
5.9.1	Other applications	33
CHAPTER 6 DATA COLLECTION		**35**
6.1	Machine Details	35
6.1.1	Details of stage 1	35
6.1.1.1	Operations performed	35
6.1.2	Details of stage 2	36
6.1.2.1	Operations performed	36
6.2	Job Data	37
CHAPTER 7 PROPOSED GA CONCEPT		**44**
7.1	Number of Parameters to be Optimized	44
7.2	Initializing Sample Population	44
7.3	Method of Representation	44
7.4	Developing Objective Function	44

7.5	Output of Reproduction	48
7.6	Method of Mutation	52
7.7	Output after the First Iteration	53
7.8	Algorithm	55
7.9	VB Program	56
CHAPTER 8 RESULTS AND ANALYSIS		**75**
8.1	Result Analysis	75
8.2	Validation	81
CHAPTER 9 CONCLUSION AND SCOPE OF FUTURE WORK		**82**
9.1	Conclusion	81
9:2	Scope of Future Work	83
REFERENCES		**84**

LIST OF TABLES

Table	Title	Page No.
1.1	Scheduling Studies	3
5.1	Chromosomes with Permutation Encoding	26
5.2	GA Applications	32
6.1	Stage 1 Machine Details	35
6.2	Stage 2 Machine Details	36
6.3	Time Studies	39
7.1	C.O.F Values for Sample Population	45
7.2	Cumulative Probability Values	47
7.3	Rank Based Selection	48
7.4	Crossover	50
7.5	Mutation	52
7.6	Output After the First Generation	54
8.1	Analysis of Results	79

LIST OF FIGURES

Figure	Title	Page No.
3.1	Plant Layout	13
4.1	Problem Solving Procedure	14
6.1	Bush	40
6.2	Setting Ring	40
6.3	Connecting Rod Rear	41
6.4	Chopper Plate	41
6.5	Bearing Support	42
6.6	Jockey Pulley	42
6.7	CNC Turning Centre	43
6.8	CNC Vertical Machining Centre	43
7.1	Input Form of the VB Program	69
7.2	Job Data Form of the VB Program	70
7.3	Calculated of objectives in VB Program	71
7.4	Reproduction Process Performed by the VB Program	72
7.5	Crossover Operation in the VB Program	73
7.6	Mutation Operation in the VB Program	74
8.1	Graph – Iteration Vs C.O.F for Pc= 0.6 & Pm = 0.01	75
8.2	Graph – Iteration Vs C.O.F for Pc= 0.6 & Pm = 0.02	75
8.3	Graph – Iteration Vs C.O.F for Pc= 0.7 & Pm = 0.01	76
8.4	Graph – Iteration Vs C.O.F for Pc= 0.7 & Pm = 0.02	76
8.5	Graph – Iteration Vs C.O.F for Pc= 0.8 & Pm = 0.01	77
8.6	Graph – Iteration Vs C.O.F for Pc= 0.8 & Pm = 0.02	77
8.7	Graph – Iteration Vs C.O.F for Pc= 0.9 & Pm = 0.01	78
8.8	Graph – Iteration Vs C.O.F for Pc= 0.9 & Pm = 0.02	78

LIST OF ABBREVATIONS

n	Number of jobs
PMS	Parallel Machine Scheduling
N	Size of Sample Population
P_c	Cross Over Rate
P_m	Mutation Rate
C.O.F	Combined Objective Function.
R	Rank
E	Expected Value
R.N	Random Number
C_j	Completion time for job j
L_j	Lateness for job j
T_j	Tardiness for job j
GA	Genetic Algorithm
TS	Tabu Search
SA	Simulated Annealing
NP	Non Polynomial

CHAPTER 1
INTRODUCTION

Scheduling problems in factory shop floors of industries are very hard to solve because of the nature of processing structure in combination with rigid technical constraints, such as no-wait restrictions. The motivation for this study originated due to various uncertainties like delay in product delivery, in effective utilization of available resources, etc. faced in flow shop by improper scheduling of jobs. In any given period, all jobs in the set of jobs have to go through a set of operations without interruption, and the set of machines in each stage. A complicated combination of objectives is used to determine the quality of a schedule, which involves minimum total makespan, earliness and tardiness.

1.1. SCOPE AND PURPOSE

This project deals with problems of scheduling a number of jobs on non-identical machines, industries are mainly interested in delivering products to customers in time and to utilize the available resources in an efficient form. Therefore makespan, earliness and tardiness parameters are introduced. The problem is intractable and consequently develops an effective heuristic to obtain near-optimal solutions. The problems of scheduling the jobs on parallel and non identical machines are considered. The goal is to find an optimal schedule, which has a minimum combined objective function consisting of minimum makespan and earliness and tardiness.

Genetic algorithm has been developed and computationally demonstrated with the help of visual basic to be efficient in obtaining near optimal solutions. Successful implementations of scheduling techniques in practice are scarce. Not only do daily disturbances lead to a gap between theory and practice, but also the extent to which a scheduling technique can adequately model the process on the shop floor and the extent to which the optimization goal that are not great enough.

1.2. SEQUENCE OF THE REPORT

This report starts with a brief introduction about scheduling, then the problem faced in the industry has been described, literature survey has been done in detail and then a methodology of Genetic Algorithm has been proposed to solve the identified problem. Then the procedure of Genetic algorithm has been described briefly and the proposed Genetic Algorithm methodology has been applied to solve the problem with the collected data from the industry. Results after application of genetic algorithm has been compared and analyzed, finally an optimized solution has been arrived.

1.3. LIMITATIONS OF THE PROJECT

This project also includes some limitations such as, machine break downs which are not being considered. There is no provision to incorporate rush and sudden orders in shop floor by providing parallel machines adjacent to key operation machine in cells. Absence of labor and tool failure is not taken in to account.

1.4 LITERATURE SURVEY

A flow shop generally consists of several different types of machines. The machines are located together to form a functional layout. The strengths of flow shops are higher levels of flexibility and utilization of resources. Some of the inherent drawbacks are longer lead-times and uncontrolled delay of certain jobs. It is also possible to remove the subjective nature of priority of jobs by assigning job priorities. The flow shop scheduling problem may be characterized as one in which a number of jobs, each containing one or more operations, requiring some amounts of time are to be processed in a particular sequence of machines. The objective is to find a processing order on each machine to optimize a selected measure of performance. In reality, industrial scheduling problems are dynamic and complex in nature. Analytical approach to flow shop problems has been proved to be extremely difficult proposition even with several restrictive assumptions. Researches and industrial practitioners have tested sequencing rules with computer simulation in flow shops in order to determine a rule or rules best suited to satisfy certain objectives for implementation.

TABLE 1.1.SCHEDULING STUDIES

AUTHOR	STUDY
Leung and Young (1989). Eck and Pinedo (1993)	Presented algorithms to minimize the makespan subject to a determined flow time level.
Gonzalez and Johnson (1980)	Proposed algorithm to minimize the makespan subject to a bound on the number of preemptions.
Shmoys and Tardos (1993)	Proposed linear combination of the makespan and a total cost function for unrelated machine models.
Sin (1989)	Considered the problem of minimizing the makespan and number of preemptions for a set of jobs restricted to due data.
Nagar (1995)	Exhaustive survey of multiple and bi-criteria scheduling.
Wilson J.M. (1985)	Alternative formulation of a flow shop scheduling problem.
Wolsey L.A. (1985)	Mixed integer programming formulations for production planning and scheduling problems.
Webster S. (1995)	Weighted flow time bounds for scheduling identical processors.
Bean J. (1994)	Genetic algorithms and random keys for sequencing and optimization.

1.4.1. Other Related Studies

During the last ten years, the scheduling problems have been intensively studied. Scheduling problems with more than one machine involves resource allocation and sequencing, rather than simply sequencing. The complexity, in general, grows exponentially, making the problems intractable. Since efficient exact algorithms have not been found in 50 years of researching, one is led to suspect the increasing appearance of these problems in computer science, besides their intractability, heuristic algorithms have been extensively developed to solve real world problems with very good results. However, for many cases, polynomial time approximation algorithms that guarantee near-optimal solutions are not known, so that it constitutes a true challenge.

Many improved algorithms have been developed for flow shop scheduling. These are based upon local search in a neighborhood. They take a feasible solution as starting-point and try to improve it by small iterative changes. This iterative improvement can be achieved by means of many different processes. Three of them are: Threshold Algorithm, Tabu Search and Genetic Algorithm. Among these, genetic algorithm starts with a set of solution instead of with only one.

Research into metaheuristics is quite extensive, especially for Job shop (Vaessens, 1994) and Flow shop (Dorn, 1996). However, applications to flow shop problems are relatively scarce. Multiple-machine scheduling are difficult problems for local optimization techniques because neighboring solutions differ widely in quality (Hubscher and Glover, 1994).

Kampke (1988) considered the PMS problem by introducing new local search techniques whose neighborhood structure is based on multiple exchanges of jobs among machines. They showed that, by means of the proposed algorithms, near optimal solutions could be obtained when the running time is not important, and satisfactory ones could be found rapidly. Jozefowska (1998) considered the discrete-continuous scheduling problem where the machines are identical and the optimization criterion is the makespan. They combined linear programming and local search methods to solve this problem. GA is preferable, finding the largest number of optimal solutions and showing deviation for all the problem sizes.

Several metaheuristics have been developed to deal with Flow shop scheduling. Van de Velde (1993) presented an algorithm based on iterative local search where the search direction is guided by surrogate multipliers. Its performance is better than all the previously published approximation algorithms. Later on Piersma and Van Dijk (1996) presented algorithms using local search which are shown to be more efficient.

Glass (1944) completed the relative performance of GA, SA and TS on the PMS problem. The performance of standard GA without incorporating some problem-specific features or another type of heuristic is poor. The algorithm obtained by combining the constructive algorithm by (Hariri and Fotts, 1991) with GA has been shown to be comparable with the SA and TS that they had developed. For these three algorithms, TS generates slightly better solutions in a short time, and GA and SA improve as the time limit increases.

Detailed literature survey shows that extensive work on flow shop scheduling has been carried out primarily for achieving the optimized solution. Large amount of heuristic rules, linear programming models and algorithms have been developed in this regard. No attempts have been made in computer based model building and search evaluation procedure, which is very important in this information technology dominated world. Multi measure performance optimization is today's need to satisfy customers as well as the objectives of the company. An attempt has been made in this project to develop computer based multi machine multi objective optimization through the emerging new concept of genetic algorithm.

CHAPTER 2
SCHEDULING

2.1. INTRODUCTION

Scheduling presents when and in what sequence the work will be done. It involves deciding as to when the work will start and in certain duration of time how much work will be finished. Scheduling deals with orders and machines, i.e., it determines which order will be taken up on which machine and in which department by which operator. While doing so, the aim is to schedule as large amount of work as the plant facilities can conveniently handle by maintaining a free flow of material along the production line.

Scheduling may be called the time phase of loading. Loading means the assignment of task or work to the facility whereas scheduling includes in addition, the specification of time and sequence in which the order/work will be taken up.

Scheduling is a temporal assignment of the orders (manufacturing products) to resources (machines) where a number of goals and conditions (meeting the due dates, using only special machines) must be regarded. Scheduling includes creating a schedule of production process (predictive scheduling) and adopting an existing schedule because of events in the scheduling environment (reactive scheduling). A Scheduling problem comprises of,

i. A set of orders to manufacture products that are to be scheduled, subjected to several constraints.
ii. A set of products with information about process plans (routings), operations, machines, etc.
iii. A set of resources with different capabilities (machines and personnel's).
iv. A set of hard constraints (production requirements) that must be fulfilled.
v. A set of soft constraints (meeting due dates) that must be fulfilled but may be relaxed.

2.2. SINGLE MACHINE SCHEDULING

Scheduling is the allocation of start and finish time to each particular order. Therefore scheduling can bring productivity in shop floor by providing a calendar for processing a set of jobs. It is nothing but scheduling various jobs on a set of resources such that certain performance measures are optimized.

The single machine scheduling problem consists of n jobs with the same single operation on each of the jobs, while the flow shop scheduling problem consists of n jobs with m operations on each of the jobs.

The basic single machine scheduling problem is characterized by the following conditions,

i. A set of independent, single-operation jobs is available for processing time at zero.
ii. Set-up time of each job is independent of its position in jobs sequence. So, the set-up time of each job can be included in its processing time.
iii. Job descriptors are known in advance.
iv. One machine is continuously available and is never kept idle when work is waiting.
v. Each job is processed till its completion without break.

The following three basic data are necessary to describe jobs in a deterministic single machine scheduling problem which are as follows.

2.2.1. Processing Time (t_j)

It is the time required to process job j. The processing time, t_j normally includes both actual processing time and set-up time.

2.2.2. Ready Time (r_j)

It is the time at which job j is available for processing. The ready time of a job is the difference between the arrival time of that job and the time at which that job is taken for processing.

2.2.3. Flow Time (F_j)

It is the amount of time job j spends in the system. Flow time is a measure which indicates the waiting time of jobs in a system. This in turn gives some idea about in-process inventory due to schedule. It is the difference between the completion time (C_j) and the ready time of the job j.

$F_j = C_j - r_j$

2.2.4. Lateness (L_j)

It is the amount of time by which the completion time of job j differs from the due date ($L_j = C_j - D_j$). Lateness is a measure which gives an idea about conformity of the jobs in a schedule to a given set of due dates of the jobs. Lateness can be either position lateness or negative lateness.

2.2.5. Tardiness (T_j)

Tardiness is the lateness of job j if it fails to meet its due date or zero, otherwise

$T_j = \max(0, C_j - d_j)$

$= \max(0, L_j)$

2.3. PARALLEL MACHINE SCHEDULING

Scheduling problems with more than one machine involve resource allocation and sequencing, rather than simply sequencing. The complexity, in general, grows exponentially, making the problems intractable. In the classical parallel machine scheduling problem (PMS), there are n jobs and m machines. Each job need to be executed on one of the machines during a fixed processing time. So the aim is to find the schedule that optimizes a certain performance measure. The scheduling process involves two kinds of decisions, sequencing (the order in which jobs are processed), and jobs machine assignment. Single machine problems ask just to find optimal job sequencing, but in the multiple machines case it is necessary to find an optimal job-machine assignment as well. The complexity usually grows exponentially with the number m of machines, making the problem intractable. This problem, like all deterministic scheduling problems, making the problem intractable. This problem, like all deterministic scheduling problems, belongs to the wide class of combinational optimization problem, many of which are known to be NP-hard, (what this means is that it is not likely that there are efficient optimization algorithms to solve them).

Many real life problems can be modeled as PMS ones. On production lines, it is common to find more than one machine of each kind carrying out the production tasks. The PMS also constitutes an important issue within the field of computer science, due to the increments in use of share time systems, or multiprocessor computers, which require efficient procedures for assigning tasks.

2.4. JOB SHOP SCHEDULING

In job shop scheduling, the production rate, delivery time etc are fixed by the sale department and limited resources have to be used effectively. In this product focused scheduling, in setting the individual type and the production rates and personal schedules, the schedule may be faced with either great rigidity or reasonable flexibility and the nature of the production system design. Scheduling a job shop is much more complex than scheduling a flow shop for three reasons:

1. Job shop handles a large variety of product, with different flow pattern through the work centers.
2. Equipment in a job shop is time shared by the various orders in process, where in a flow shop it is used exclusively for the product type.
3. Different jobs may be governed by different priorities. This in turn affects the orders I which they are selected for processing once they are assigned to a work centre. Thus uniformity of output produced for stock by a flow shop, however does not create such problem. Order priorities for flow shop affect mainly their shipping rather than processing dates.

The goals of job shop scheduling usually are,

- High percentage of orders completed on time.
- High utilization of workers and facilities.
- Low in process inventory
- Low overtime.

Job shop scheduling applies to intermittent operations of all types, whether in a factory, hospital, courtroom, or restaurant. For service operations, the form "job" is replaced by patient, customer, client or whatever flows through the facility. The work center may be desk, office, room, or skill specialty.

2.5. FLOW SHOP SCHEDULING

In flow shop scheduling problem, there are n jobs, each requires processing on m different machines. The order in which the machines are required to process a job is called process sequence of that job. The process sequences of all the jobs are the same. But the processing times for various jobs on a machine may differ. If an operation of that job is assumed as zero. The flow shop scheduling problem can be characterized as given below,

i. A set of multiple-operation jobs is available for processing at time zero (Each job requires m operations and each operations requires a different machines.
ii. Set-up times for the operations are sequence independent and are included in processing times.
iii. Job descriptors are known in advance.
iv. 'm' different machines are continuously available.
v. Each individual operation of jobs is processed till its completion without break.

CHAPTER 3

OBJECTIVES AND PROBLEM IDENTIFIFCATION

3.1. OBJECTIVES

In recent days, every industry faces tough competition in manufacturing and selling their products. Each and every product being better quality, the one which has competitive (low) price, wins the product.

The company chosen is involved in manufacturing work. Scheduling is a superior problem in flow shop industry. The aim of this project is to arrive at an alternative sequence of scheduling with an objective to reduce the manufacturing time, there by reducing manufacturing cost of the products.

The main objectives are

i. Makespan minimization
ii. To reduce earliness
iii. To reduce tardiness
iv. Prompt product delivery
v. Effective utilization of resources

3.2. PROBLEM IDENTIFICATION

The company supplies industrial components to variety of customers. It schedules jobs according to priority of jobs to be given to the customers, but it does not follow any organized procedure for scheduling the jobs. Due to this, the company is facing difficulties of resources etc. particularly, when the order is more, difficulties become more intense.

Selection of a particular system depends on a number of factors. Some of these are given below:

i. Available infrastructure like machine capacity and number of machines.
ii. Volume of manufacturing like low volume or high volume.

iii. Priority of jobs.

iv. Time schedules like operation, waiting and delivery time.

Each system of scheduling has its merits and demerits. Hence there is nothing like an ideal system or a best system for scheduling. What is relevant is to select the most appropriate system of scheduling which suits a particular manufacturing set up. This ability to select the most appropriate system depends on the professional and conceptual skill of manufacturing managers. Thus scheduling is a dynamic system.

3.3 PROBLEM DEFINATION

In this study, there are about 2 stages of machines. In each stage similar machining operations are performed. In stage 1, the machines available are two CNC turning center and one Conventional turning center. In stage 2, the machines available are two Conventional VMC and one CNC VMC. 10 jobs are being considered for scheduling. Each job varies in its dimensions, requiring change in fixture setup and tools.

10 jobs are being processed in a sequence, according to priority of jobs to be given to customers. While processing in those sequences, organization is facing difficulty in delivering the products within due dates and available resources are not properly utilized. So these 10 jobs are to be scheduled properly, so that the objectives could be met.

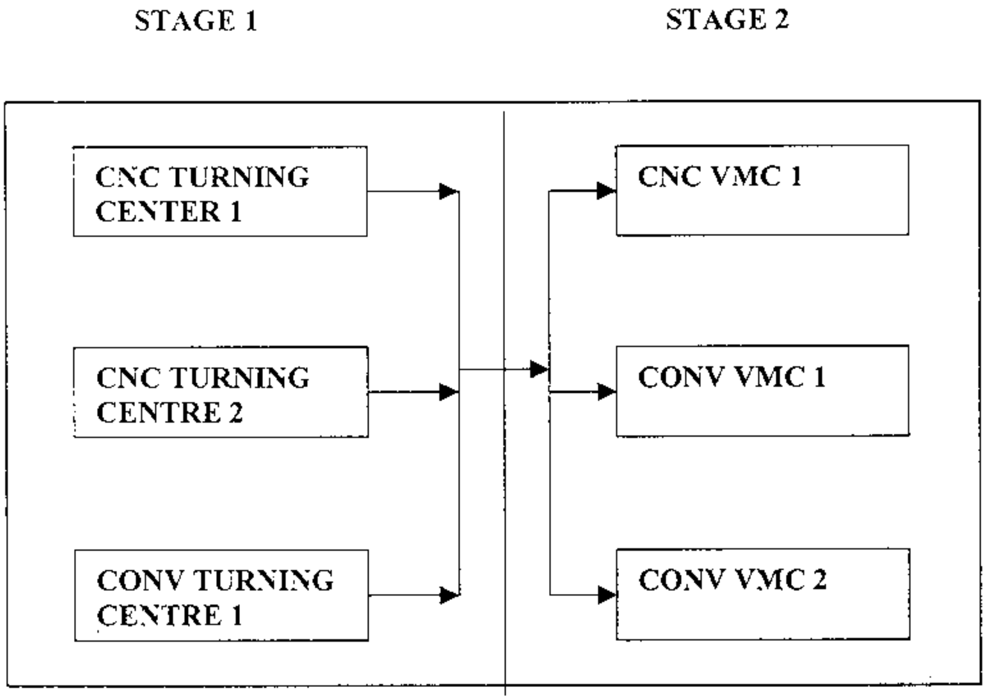


FIG.3.1. PLANT LAYOUT

The following assumptions are usually made while dealing with sequencing problems:

i. Only one operation is carried out on a machine at a particular time.
ii. Each operation, once started, must be completed.
iii. An operation must be completed before its succeeding operation can start.
iv. A job is processed as soon as possible, but only in the order specified.
v. Processing times are independent of order of performing the operations.
vi. The transportation time i.e., the time required to transport jobs from one machine to another is negligible
vii. Jobs are completely known and are ready for processing when the period under consideration starts.

CHAPTER 4
PROBLEM SOLVING

4.1 PROBLEM SOLVING PROCEDURE

The proposed problem has been solved by following the below mentioned figure 4.1,

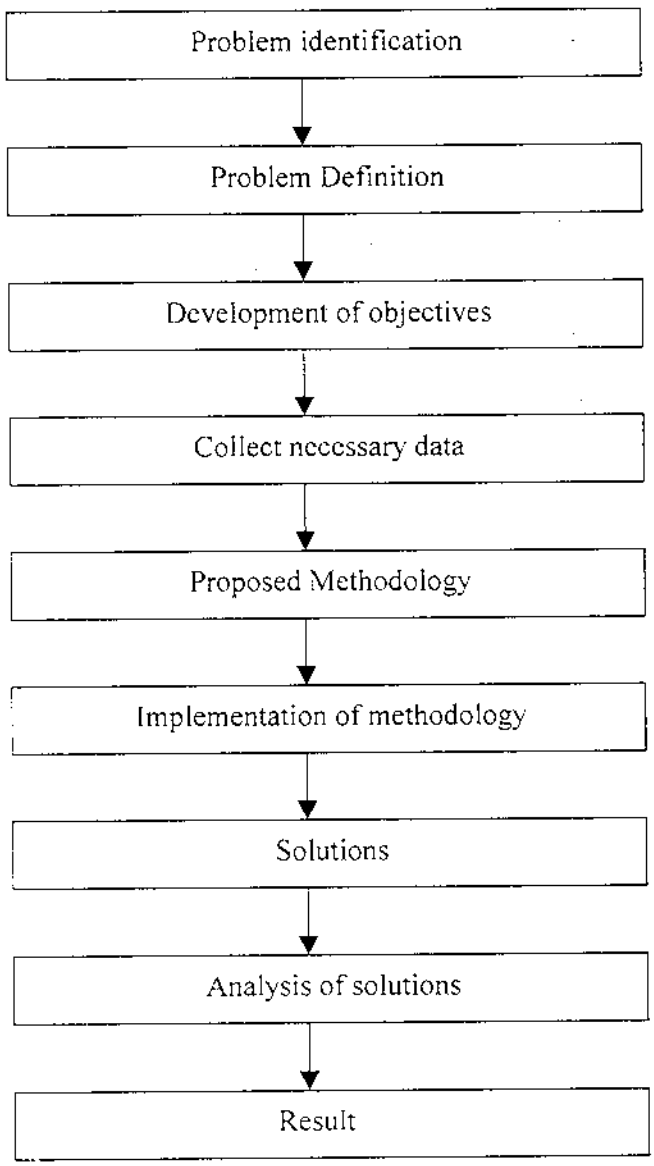


FIG.4.1. PROBLEM SOLVING PROCEDURE

4.2. PROBLEM FORMULATION:

4.2.1 Earliness and Tardiness

Earliness or Tardiness of single job = Due time – completion time of that job (min).

If Due time > Completion time = Earliness of a job

If Due time< Completion time = Tardiness of a job

Total Earliness = sum of the earliness of the jobs in the sequence.

Total Tardiness = sum of the Tardiness of the jobs in the sequence.

4.2.2 Makespan

Make Span is the total completion time of the jobs in that specified sequence. That is the maximum time taken for the completion of production of all jobs.

4.2.3 Combined Objective Function (COF)

C.O.F = W1 *(Make span of a sequence / Average Make span) +
W2*(Total Earliness of a sequence / Average Earliness) +
W3*(Total Tardiness of a sequence / Tardiness Average)

Where W1= Weightage for Make span
W2=Weightage for Total Earliness
W3=Weightage for Total Tardiness

4.3 PROPOSED METHODOLOGY:

The optimization process is done by using the algorithm called Genetic Algorithm. Here an initial population of 30 sequences is randomly taken from the huge population of sequences. From this initial population the best sequences are extracted by rank selection. Thus the filtered sequences are reproduced. After reproduction, cross over is done from parent sequences. After that mutation is done to get the required child sequence.

4.4 PROCEDURE FOR APPLYING GENETIC ALGORITHM:

4.4.1 Input for Algorithm

The inputs like.

i. No of CNC and conventional machines
ii. No of jobs
iii. CNC machining time
iv. Conventional machining time
v. Weightage for Make span, Earliness and tardiness.
vi. Cross over and Mutation probability, Number of iterations etc, are given

4.4.2 Genetic Algorithm Representation

A basic GA can be represented in following steps:

Step 1: Represent the problem variable domain as a chromosome of a fixed length, choose the size of a chromosome population N, the crossover probability ***pc*** and he mutation probability ***pm.***

Step 2: Define a fitness function tom measure the performance, or fitness of an individual chromosome in the problem domain. The fitness function establishes the basis for selection of chromosomes that will be mated together during reproduction.

Step 3: Randomly generate an initial population of chromosomes of size N: X1, X2...Xn

Step 4: Calculate the fitness of each individual chromosome: f(x1), f(x2)... f (xn).

Step 5: Select a pair of chromosomes for mating from the current population, parent chromosomes are selected with a probability related to their fitness. Highly fit chromosomes have a higher probability of being selected for mating than less fit chromosomes.

Step 6: Create a pair of offspring chromosomes by applying genetic operator`s crossover and mutation.

Step 7: Place the created offspring chromosomes in the new population.

Step 8: Repeat step 5 until the size of new chromosome population becomes equal to the size of the initial population N.

Step 9: Replace the initial (parent) chromosome population with the new (off spring) population.

Step 10: Go to step 4, and repeat the process until the termination criterion is satisfied.

GAs represents an iterative process. Each iteration is called a generation. A typical number of generations for a simple GA can vary from 50 to over 500[10].the entire set of generations is called a run. We expect to find one or more highly fit chromosomes.

CHAPTER 5
METHODOLOGY

Many real world engineering problems are likely to be a multi model with many local optimum points. Conventional optimization procedures may easily get trapped in any one of the local optimum points.. The only way to solve the above problem for global optimality is to have a starting point in the global basin. Genetic algorithm use search in descent and ascent direction by using probability in all their operators. Since an initial random population is used, genetic algorithms are ideally suited for problems of global optimization moreover, in case of scheduling n jobs there are n! ways of doing it, hence it is a difficult task to arrive an optimal solution in this very large search space traditionally, so a non traditional methodology genetic algorithm is being used to achieve the global optimality.

5.1. GENETIC ALGORITHM

A genetic algorithm (GA) is a search technique used in computer science to find approximate solutions to optimization and search problems. Genetic algorithms are a particular class of evolutionary algorithms that use techniques inspired by evolutionary biology such as inheritance, mutation, natural selection, and recombination (or crossover).

Genetic algorithms are typically implemented as a computer simulation in which a population of abstract representations (called chromosomes) of candidate solutions (called individuals) to an optimization problem evolves toward better solutions. Traditionally, solutions are represented in binary as strings of 0s and 1s, but different encodings are also possible. The evolution starts from a population of completely random individuals and happens in generations.In each generation, the fitness of the whole population is evaluated, multiple individuals are stochastically selected from the current population (based on their fitness), modified (mutated or recombined) to form a new population, which becomes current in the next iteration of the algorithm

5.1.1. History Of Genetic Algorithm

Genetic algorithms originated from the studies of cellular automation, conducted by John Holland and his colleagues at the University of Michigan. Research in GAs remained largely theoretical until the mid-1980s, when The First International Conference on Genetic Algorithms was held at The University of Illinois. As academic interest grew, the dramatic increase in desktop computational power allowed for practical application of the new technique. In 1989, The New York Times writer John Markoff wrote about Evolver, the first commercially available desktop genetic algorithm. Custom computer applications began to emerge in a wide variety of fields, and these algorithms are now used by a majority of Fortune 500 companies to solve difficult scheduling, data fitting, trend spotting and budgeting problems, and virtually any other type of combinatorial optimization problem.

5.1.2 Evolution

John Holland, from the University of Michigan began his work on genetic algorithm at the beginning of the 60s. A first achievement was the publication of Adaptation in Natural and Artificial System in 1975

Holland had a double aim:

i. To improve the understanding of natural adaptation process,

ii. To design artificial systems having properties similar to natural systems

The basic idea is as follow: the genetic pool of a given population potentially contains the solution, or a better solution, to a given adaptive problem. This solution is not "active" because the genetic combination on which it relies is split between several subjects. Only the association of different genomes can lead to the solution. Holland method is especially effective because he not only considered the role of mutation (mutations improve very seldom the algorithms), but he also utilized genetic recombination, (crossover) these recombination, the crossover of partial solutions greatly improve the capability of the algorithm to approach, and eventually find, the optimum.

5.1.3. Basic Concepts of GA

Genetic algorithm is good at taking larger, potentially huge, search spaces and navigating them looking for optimal combinations of things and solutions which we might not find in a life time.

To use a genetic algorithm, we must represent a solution to our problem as a genome (or chromosome). The genetic algorithm then creates a population of solutions and applies genetic operators such as mutation and crossover to evolve the solutions in order to find the best one(s).

This presentation outlines some of the basics of genetic algorithms. The three most important aspects of using genetic algorithms are:

i. Definition of the objective function,
ii. Definition and implementation of the genetic representation,
iii. Definition and implementation of the genetic operators.

Once these three have been defined, the generic genetic algorithm should work fairly well. Beyond that you can try many different variations to improve performance, find multiple optima (species - if they exist), or parallelize the algorithms.

Genetic algorithm should be used in case,

i. alternate solutions are too slow or overly complicated
ii. need an exploratory tool to examine new approaches,
iii. problem is similar to one that has already been successfully solved by using GA
iv. we want to hybridize with an existing solution
v. Benefits of GA technology meet key problem requirements.

5.1.4. Differences and Similarities between GA and Other Traditional Methods

GA differs from conventional optimization and search procedure in several procedures in several fundamental ways:

i. Genetic algorithms work with coding solution set not the solution themselves.
ii. Genetic algorithms search from a population of solutions, not a single solution.
iii. Genetic algorithms use fitness function, not derivatives or other auxiliary knowledge
iv. Genetic algorithms use probabilistic transition roles, not deterministic rules.

Even though GAs, are different than most traditional search algorithms, there are some similarities. In traditional search methods, where a search direction is used to find a new point, at least two points are either implicitly or explicitly used to define the search direction. In the cross over operator, two points are used to create new points. Thus, cross over operator is similar to a directional search method with an exception that the search direction is not fixed for all points in the population and that no effort is made to find the optimal point in any particular direction. Since two points used in cross over operator are chosen at random, many search directions are possible. Among them, some may lead to global basin and some may not. The reproduction operator has an indirect effect of filtering the good search direction and helps to guide the search. The purpose of mutation operator is to create a point in the vicinity of the current point. The search in the mutation operator is similar to a local search method such as exploratory search used in Hooke-jeeves method.

i. Population (chromosomes)
ii. Evaluation (fitness)
iii. Selection (mating pool)
iv. Genetic operators

5.2 STRUCTURE OF GENETIC ALGORITHM

The genetic algorithms usual form was described by Goldberg (1989). Genetic algorithms, differing from conventional techniques, start with an initial set of random solution called population.

Each individual in the population is called a chromosome, representing a solution. A chromosome is a string of symbols. It is usually, but not necessarily, a binary bit string. The chromosome evolve through successive iterations, called generation, the chromosome are evaluated, using some measure of fitness.

To create the next generation, new chromosomes called offspring are formed either,

i. Merging two chromosomes from current generation using a cross over operator.
ii. Modifying a chromosome using a mutation operator.

A new generation is formed by:

i. Selecting, according to the fitness values, some of the parents offspring and
ii. Rejecting others so as to keep the population size constant.

The chromosomes having higher probability will be selected. The chromosomes having lower probability will die out. So after many generations, the method will converge to an optimum or suboptimum solution.

There are two kinds of operations in genetic algorithms.

i. Evolution operation: selection
ii. Genetic operation: crossover and mutation

The genetic algorithm mimics the process of heredity of genes to create new offspring at each generation. The evolution operation mimics the process of Darwinian evolution to create populations from generation to generations.

5.3. BASIC TERMS OF GA

5.3.1 Chromosome

In genetic algorithms, a chromosome (also sometimes called a genome) is a set of parameters which define a proposed solution to the problem that the genetic algorithm is trying to solve. The chromosome is often represented as a simple string; although a wide variety of other data structures are also in use as chromosomes.

A genetic algorithm creates many chromosomes, either randomly or by design, as an initial population. These chromosomes are each evaluated by the fitness function, which ranks them according to how good their solution is. The chromosomes which produced the best solutions, relatively speaking within the population, are allowed to breed, called crossover. The best chromosomes' data is mixed, hopefully producing a better next generation.

5.3.2. Canditate Solution

In optimization (a branch of mathematics), a candidate solution is a member of a set of possible solutions to a given problem. A candidate solution does not have to be a likely or reasonable solution to the problem. The space of all candidate solutions is called the feasible region or the feasible area or solution space.

In the case of the genetic algorithm, the candidate solutions are the individuals in the population being evolved by the algorithm.

5.3.3. Fitness

In optimization techniques an objective measure is how good the solutions it finds are, e.g. a way of building a bridge across the M4 will cost 400,000. In genetic algorithms and genetic programming, by analogy with natural selection this is called fitness. Fitness is used to guide the search by deciding which individuals will be used as future points to look for better solutions

5.3.4. Fitness Function

This is a type of objective function that quantifies the optimality of a solution (that is, a chromosome) in a genetic algorithm. So that particular chromosome may be ranked against all the other chromosomes. Optimal chromosomes, or at least chromosomes which are more optimal, are allowed to breed and mix their datasets by any of several techniques, producing a new generation that will (hopefully) be even better.

Another way of looking at fitness functions is in terms of a fitness landscape, which shows the fitness for each possible chromosome.

An ideal fitness function correlates closely with the algorithm's goal, and yet may be computed quickly. Speed of execution is very important, as a typical genetic algorithm must be iterated many, many times in order to produce a useable result for a non-trivial problem.

5.4. CONTROL PARAMETERS

The efficiency of a GA is highly dependent on the values of the algorithm's control parameters. Assuming that basic features like the selection procedure are predetermined, the control parameters available for adjustment are,

i. The population size N,
ii. The crossover probability P_c, and
iii. The mutation probability P_m

5.5. COMPONENTS OF GA

5.5.1. Population Size

Population size determines how many chromosomes are in a given population (in one generation). If there are too few chromosomes, the possibility of performing crossover is reduced and only a small part of search space is explored. On the other hand, if there are too many chromosomes, then the whole process of GA is slowed down. Research shows that after some limit (which depends mainly on encoding and the problem) it is not useful to use very large populations because it does not solve the problem faster than moderate sized populations.

5.5.2. Crossover Probability:

If there is no crossover, offspring are exact copies of parents. If there is crossover, offspring are made from parts of both parent's chromosome. If crossover probability is **100%**, then all offspring are made by crossover. If it is **0%**, whole new generation is made from exact copies of chromosomes from old population

Crossover is made in hope that new chromosomes will contain good parts of old chromosomes and therefore the new chromosomes will be better. However, it is good to leave some part of old populations survive to next generation.

5.5.3. Mutation Probability:

If there is no mutation, offspring are generated immediately after crossover (or directly copied) without any change. If mutation is performed, one or more parts of a chromosome are changed. If mutation probability is **100%**, whole chromosome is changed, if it is **0%**, nothing is changed.

Mutation generally prevents the GA from falling into local extremes. Mutation should not occur very often, because then GA will in fact change to **random search**.

5.5.4. Encoding

There are many ways of representing individual genes. Holland (1975) worked mainly with string bits but we can use arrays, trees, lists or any other object. Various methods of encoding are.

i. Binary coding
ii. Octal coding
iii. Hexadecimal coding

iv. Permutation encoding

v. Value encoding

vi. Tree encoding

The type of encoding used here is Permutation encoding.

5.5.4.1. Permutation Encoding

Permutation encoding can be used in ordering problems, such as traveling salesman problem or task ordering problem.

In permutation encoding, every chromosome is a string of numbers that represent a position in a sequence.

Chromosome A	1 5 3 2 6 4 7 9 8
Chromosome B	8 5 6 7 2 3 1 4 9

TABLE .5.1. CHROMOSOMES WITH PERMUTATION ENCODING

Permutation encoding is useful for ordering problems. For some types of crossover and mutation corrections must be made to leave the chromosome consistent (i.e. have real sequence in it) for some problems.

5.6. GENETIC OPERATORS

A genetic operator is a process used in genetic algorithms to maintain genetic diversity. Genetic variation is a necessity for the process of evolution. Genetic operators used in genetic algorithms are analogous to those which occur in the natural world: survival of the fittest, or selection; asexual or sexual reproduction (crossover, or recombination); and mutation

5.6.1 Reproduction

Reproduction is usually the first operator applied on population. Chromosomes are selected from the population to be parents to cross over and produce offspring. According to Darwin's evolution theory of survival of the fittest, the best ones should survive and create new offspring. That is why reproduction operator is sometimes known as the selection operator. These exists a number of reproduction operators in GA literature but the essential idea in all of them is that the above average strings are picked from the current population and their multiple copies are inserted in the mating pool in a probabilistic manner. The various methods of selecting chromosomes for parents to cross over are:

i. Roulette-wheel selection
ii. Boltzmann selection
iii. Tournament selection
iv. Rank selection
v. Steady-state selection

Type of selection used is Rank selection.

5.6.1.1. Rank Selection

Rank selection ranks the population first and then every chromosome receives fitness value determined by this ranking The worst will have the fitness **1**, the second worst **2** etc. and the best will have fitness **N** (number of chromosomes in population).

5.6.2. Crossover

The chromosomes of the parent are mixed in some way during crossover, typically by simply swapping a portion of the underlying data structure. This process is repeated with different parent organisms until there are an appropriate number of candidate solutions in the second generation poll. There exit many types of crossover operations in genetic algorithm.

5.6.2.1. One Point Crossover

In a single-site crossover, a cross-site is selected randomly along the length of the mated strings and bits next to the cross-sites are exchanged. If an appropriate site is chosen, a better can be obtained by combining good substance of parents. Since the knowledge of the appropriate site is not known and it is not known and it selected randomly, this random selection of cross-site may produce enhanced children if the selected site is appropriate. If not, it may severely hamper the string quality. Anyway, because of the crossing of patent better children are produced and that will continue in next generation also. But if good strings are not created by crossover, they will not survive beyond next generation because reproduction will not select those strings for the next mating pool.

5.6.2.2. Two-Point Crossover

In a two-point operator, two random sites are chosen and the contents bracketed by these sites are exchanged between two mated parents.

5.6.2.3. Multi-Point Crossover

In a multi-point crossover, again there are two cases. One is even number of cross-sites and second one is the odd number of cross-sites. In case of even numbered cross-sites, the string is treated as a ring with no beginning or end. The cross-sites are selected around the circle uniformly at random. Now information between alternate pairs of sites is interchanged. If the number of cross-sites is odd, then a different cross-point is always assumed at the string beginning. The information between alternate pairs is exchanged.

5.6.3. Mutation

In genetic algorithms, mutation is a genetic operator used to maintain genetic diversity from one generation of a population of chromosomes to the next. It is analogous to biological mutation.

A classic example of a mutation operator involves a probability that an arbitrary bit in a genetic sequence will be changed from its original state. A common method of implementing the mutation operator involves generating a random variable for each bit in a sequence. This random variable tells whether or not a particular bit will be modified

5.7. OPERATION OF GA

Two elements are required for any problem before a genetic algorithm can be used to search for a solution: First, there must be a method of representing a solution in a manner that can be manipulated by the algorithm. Traditionally, a solution can be represented by a string of bits, numbers or characters. Second, there must be some method of measuring the quality of any proposed solution, using a fitness function.

For instance, if the problem involves fitting as many different weights as possible into a knapsack without breaking it, a representation of a solution might be a string of bits, where each bit represents a different weight, and the value of the bit (0 or 1) represents whether or not the weight is added to the knapsack. The fitness of the solution would be measured by determining the total weight of the proposed solution: The higher the weight, the greater the fitness, provided that the solution is possible

5.7.1. Initialization

Initially many individual solutions are randomly generated to form an initial population. The population size depends on the nature of the problem, but typically contains several hundreds or thousands of possible solutions. Traditionally, the population is generated randomly, covering the entire range of possible solutions (the search space). Occasionally, the solutions may be "seeded" in areas where optimal solutions are likely to be found.

5.7.2. Selection

During each successive epoch, a proportion of the existing population is selected to breed a new generation. Individual solutions are selected through a fitness-based process, where fitter solutions (as measured by a fitness function) are typically more likely to be selected. Certain selection methods rate the fitness of each solution and preferentially select the best solutions. Other methods rate only a random sample of the population, as this process may be very time-consuming.

Most functions are stochastic and designed so that a small proportion of less fit solutions are selected. This helps keep the diversity of the population large, preventing premature convergence on poor solutions. Here Rank selection method is used. There are several generic selection algorithms. There are several generic selection algorithms. One of the common ones is the so-called rank selection, which can be implemented as follows: The fitness function is evaluated for each individual.

a) The population is sorted by descending fitness values
b) Accumulated fitness values are computed
c) A random number R between 0 and 1 is chosen.

5.7.3. Reproduction

The next step is to generate a second generation population of solutions from those selected through genetic operators: crossover (or recombination), and mutation.

For each new solution to be produced, a pair of "parent" solutions is selected for breeding from the pool selected previously. By producing a "child" solution using the methods of crossover and mutation.

5.7.4.Crossover

This operation is performed upon the selected chromosomes. Most genetic algorithms will have a single tweakable probability of crossover (P_c), typically between 0.6 and 1.0, which encodes the probability that two selected organisms will actually breed. A random number between 0 and 1 is generated, and if it falls under the crossover threshold, the organisms are mated; otherwise, they are propagated into the next generation unchanged. Crossover results in two new child chromosomes. which are added to the second generation pool. The chromosomes of the parents are mixed in some way during crossover. typically by simply swapping a portion of the underlying data structure (although other, more complex merging mechanisms have proved useful for certain types of problems.) This process is repeated with different parent organisms until there are an appropriate number of candidate solutions in the second generation pool.

5.7.5. Mutation

The next step is to mutate the newly created offspring. Typical genetic algorithms have fixed, very small probability of mutation (P_m) of perhaps 0.01 or less. A random number between 0 and 1 is generated; if it falls within the P_m range, the new child organism's chromosome is randomly mutated in some way, typically by simple randomly altering bits in chromosome data structure. Mutation server as important role in genetic algorithm,

i. Replacing the genes lost from the population during the selection process so that they can be tried in a new context.
ii. Providing the genes that were not present in the initial population.

5.8. ADVANTAGES OF GENETIC ALGORITHM

5.8.1. Population Based Search

For optimization problem a deterministic is been generated for the consumption based on the gradient or higher order destination of objective function. This method is applied to a single point in search space. This proceeds in an increasing or decreasing order of direction search. This makes it to escape from local optima. By this it reproduces relating a god solution and the bad solution get died and produces an optimal result.

5.8.2. Major Advantages

Genetic algorithms have received considerable attention regarding their potential as a novel optimization technique. There are three major advantages when applying genetic algorithm to optimization problems.

i. Genetic algorithms do not have much mathematical requirements about the optimization problem. Due to their evolutionary nature, genetic algorithms will search for solution without regard to the specific inner workings of the problem. Genetic algorithms can handle any kind of objective function and much kind of constraints (linear or non-linear) defined on discrete, continuous, or mixed search space.

ii. The ergodicity of evolution operators makes genetic algorithms very effective at performing global search (in probability). The traditional approaches perform local search by a convergent stepwise procedure, which compares the values of nearby points and moves to the problem possesses certain convexity property that essentially guarantee that any local optima is a global optima.

iii. Genetic algorithms provide us a great flexibility to hybridize with domain dependent heuristic to make an efficient implementation for a specific problem.

5.9. GA APPLICATIONS

TABLE 5.2. GA APPLICATIONS

Domains	Application types
Control	Gas pipe line, pole balancing, missile evasion and pursuit.
Design	Semi conductor layout, aircraft design, keyboard configuration and communication networks.
Scheduling	Manufacturing, facility scheduling and resource allocation

Robotics	Trajectory planning
Machine learning	Designing neural networks and classification algorithms.
Signal processing	Filter design
Game playing	Poker, checker and prisoner's dilemma
Combinatorial optimization	Set covering, traveling salesman, routing, bin packing, graph coloring and partitioning.

5.9.1. Other Applications:

- Automated design, including research on composite material design and multi-objective design of automotive components for crashworthiness, weight savings, and other characteristics.
- Automated design of mechatronic systems using bond graphs and genetic programming (NSF).
- Calculation of Bound States and Local Density Approximations
- Configuration applications, particularly physics applications of optimal molecule configurations for particular systems like C60 (buckyballs).
- Container loading optimization.
- Distributed computer network topologies.
- File allocation for a distributed system.
- Parallelization of GAs/GPs including use of hierarchical decomposition of problem domains and design spaces nesting of irregular shapes using feature matching and GAs.
- Game Theory Equilibrium Resolution
- Learning Robot behavior using Genetic Algorithms.
- Mobile communications infrastructure optimization.
- Molecular Structure Optimization (Chemistry)
- Multiple population topologies and interchange methodologies.
- Protein folding and protein/ligand docking.
- Plant floor layout.

- Scheduling applications, including job-shop scheduling. The objective being to schedule jobs in a sequence dependent or non-sequence dependent setup environment for a minimal total tardiness.
- Solving the machine-component grouping problem required for cellular manufacturing systems
- Stock Market Prediction
- Tactical asset allocation and international equity strategies.
- Traveling Salesman Problem.

CHAPTER 6

DATA COLLECTION

6.1. MACHINE DETAILS

6.1.1. Details of Stage 1

TABLE 6.1 STAGE 1 MACHINE DETAILS

No	Machine number	Machine type
1	MC01	ACE jobber XL CNC turning centre
2	MC02	ACE jobber XL CNC turning centre
3	MC03	lathe

6.1.1.1. Operations Performed

- Facing
- Turning
- Boring
- Parting
- Grooving
- Threading

6.1.2. Details of Stage 2

TABLE 6.2 STAGE 2 MACHINE DETAILS

No	Machine number	Machine type
1	MC01	CNC VMC
2	MC02	VERTICAL MILLING MACHINE
3	MC03	VERTICAL MILLING MACHINE

6.1.2.1. Operations Performed

- Side milling
- Arc milling
- Slot milling
- Flat milling
- Slotting
- Square milling

6.2. JOB DATA

Job 1: Bearing collar

Application: hydraulic pump

Operation 1: facing, turning, boring

Operation 2: slot milling

Job 2: Bush

Application: textile spares

Operation 1: facing, turning, grooving, parting

Operation 2: facing, boring

Material: EN8 Steel

Job 3: Setting ring

Application: textiles

Operation 1: facing, turning

Operation 2: facing, turning, boring

Material: Cast Iron

Job 4: Connecting rod rear

Application: hydraulics

Operation1: boring, grooving

Operation2: reaming, face milling, slotting, spot milling

Material: Cast Iron

Job 5: Support

Application: hydraulics

Operation1: facing, boring

Operation2: side milling, drilling, tapping

Material: Cast Iron

Job 6: Jockey pulley

Application: textile

Operation1: turning, boring

Operation2: arc milling

Material: Tin Sheet

Job 7: Disc

Operation1: facing, outer diameter turning, threading

Operation2: slot milling

Material: Aluminium

Job 8: Chopper plate

Application: food processing industries

Operation1: drilling, radial milling

Operation2: facing, outer diameter turning

Material: Stainless Steel

Job 9: Radial ring

Application:

Operation1: outer diameter turning

Operation2: square milling

Material: Aluminium

Job 10: bearing support

Application:

Operation1: facing, boring

Operation2: boring, side milling

Material: EN8 Steel

TABLE 6.3 TIME STUDIES

Sl.no.	Job	Quantity	Due date (days)	Time for stage1 (Iin mins)		Time for stage2(in mins)	
				CNC T.C	CONV. LATHE	CNC VMC	CONV. VMC
1	Bearing collar	190	3	03	15	02	05
2	Bush	155	2	03	13	04	10
3	Setting ring	100	3	03	11	01	05
4	Connecting rod	215	1	03	10	25	90
5	Support	185	3	03	15	12	75
6	Jockey pulley	120	3	01	15	02	12
7	Disc	150	1	03	15	02	05
8	Chopper plate	100	2	03	10	06	60
9	Radial ring	150	3	01	03	02	10
10	Bearing support	275	2	03	02	04	45

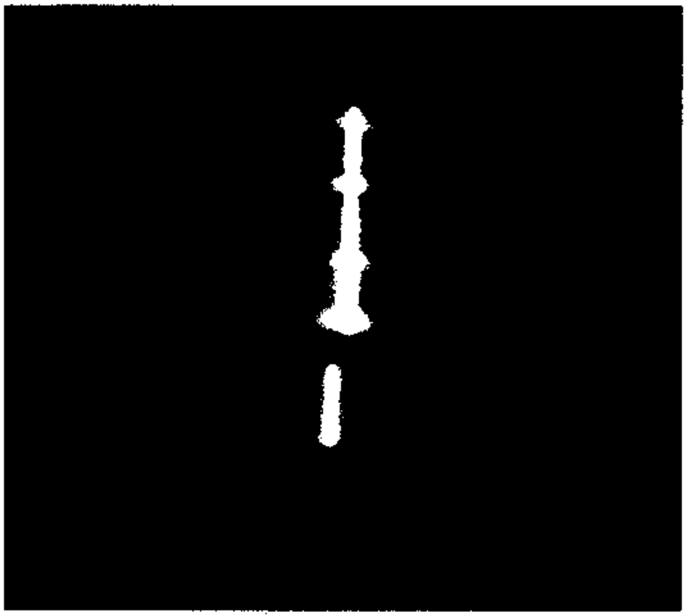


Fig.6.1. BUSH

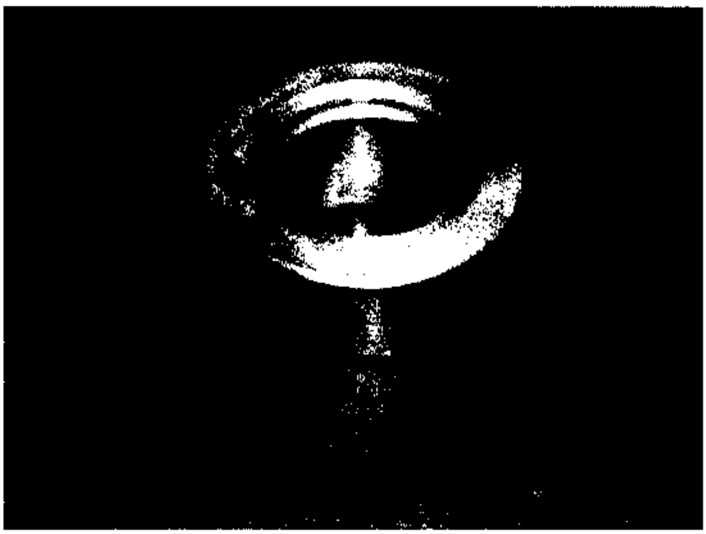


Fig.6.2. SETTING RING

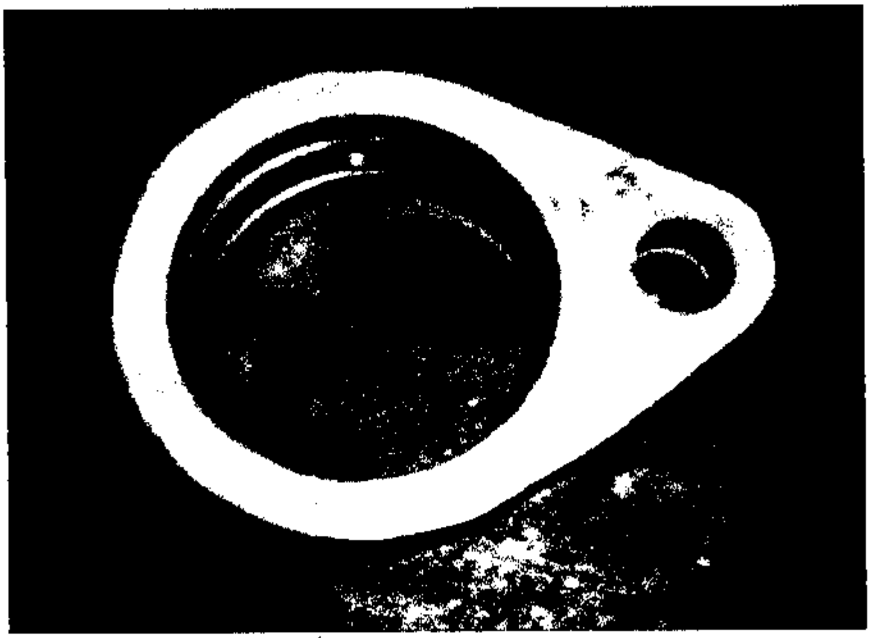


Fig.6.3. CONNECTING ROD REAR

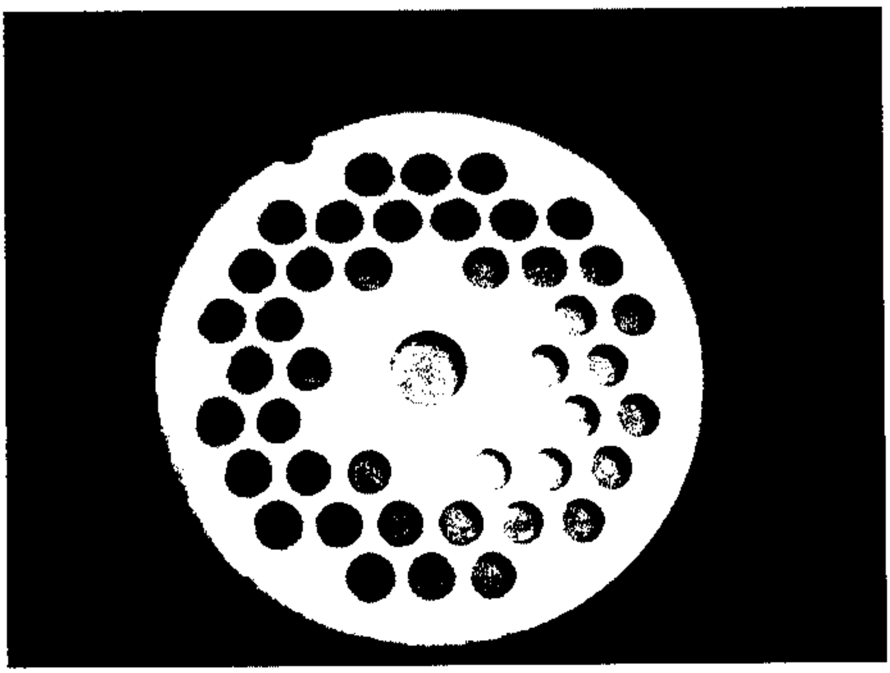


Fig.6.4.CHOPPER PLATE

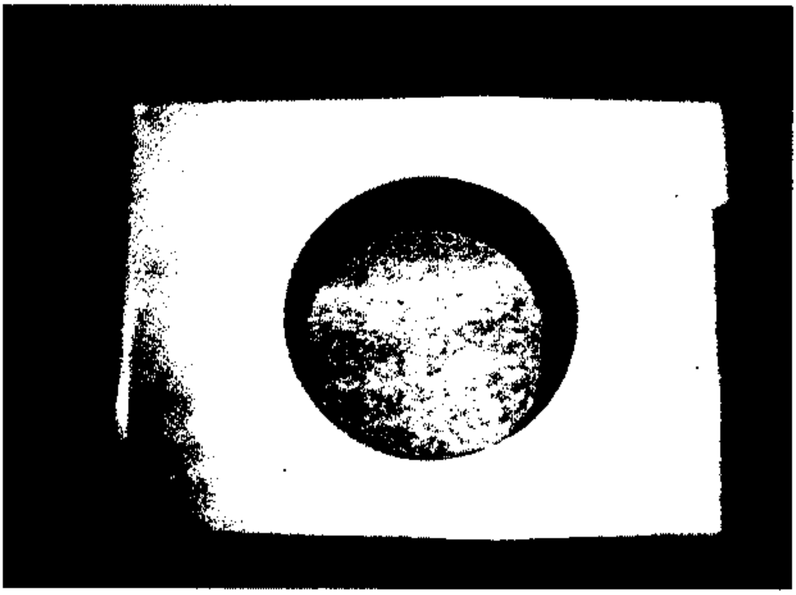


Fig.6.5. BEARING SUPPORT

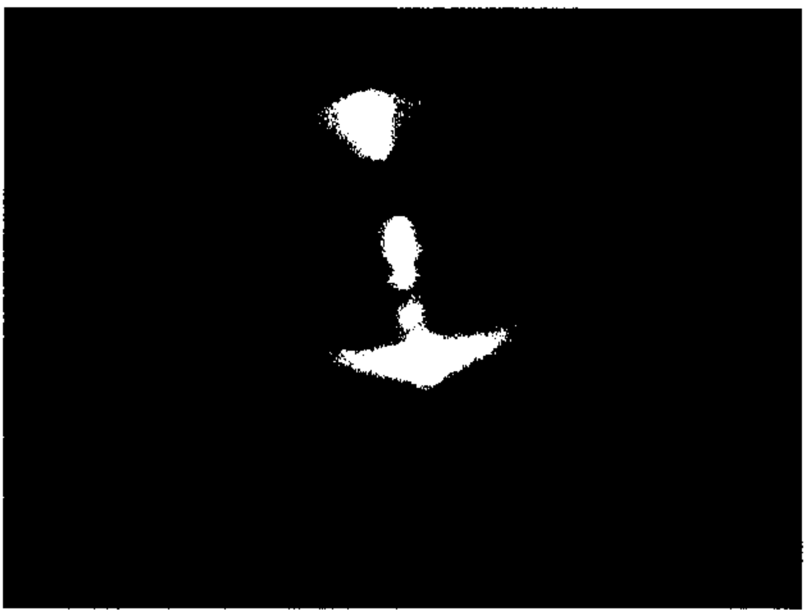


Fig.6.6. JACKEY PULLEY

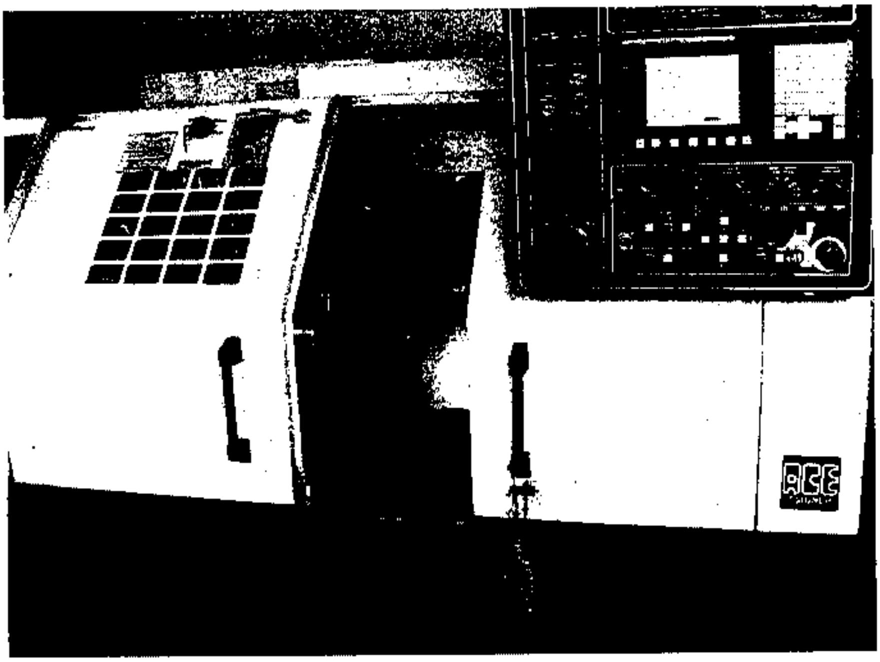


Fig.6.7. CNC TURNING CENTRE

Fig.6.8. CNC VERTICAL MACHINING CENTRE

CHAPTER 7

PROPOSED GENETIC ALGORITHM CONCEPT

7.1. NUMBER OF PARAMETERS TO BE OPTIMIZED

In this problem, there are three parameters to be optimized

i. Total Makespan
ii. Earliness
iii. Tardiness

7.2. INITIALIZING SAMPLE POPULATION

Sample population is being initialized according to requirement. Size of sample population is defined as N, where N must be an even number so that the pairs could be made to reproduce. In this study sample population is 30.
Efficiency of GA is highly dependent on value of population size N. So population size acts as a control parameter.

7.3. METHOD OF REPRESENTATION

There are so many methods of representation. But in this study each schedule has been represented in form of integers from 1 to 10, as there are 10 jobs considered in this problem. Moreover, using integers to represent the schedule, it serves as an easy method for computational work.
For example, each schedule is represented as 1-2-3-4-5-6-7-8-9-10. Total makespan, earliness and tardiness are calculated for each schedule.

7.4 DEVELOPING OBJECTIVE FUNCTION

Objective function has been developed for the proposed problem, as the problem involves multi- objectives, combined objective function has been developed using the following formulae.

C.O.F = w1*(Makespan / Average of makespan) + w2*(Earliness / Average earliness) + w3*(Tardiness/ Average tardiness)

w1= Weightage factor for total makespan

w2= Weightage factor for total earliness

w3= Weightage factor for tardiness respectively

TABLE7.1. C.O.F VALUES FOR SAMPLE POPULATION

sno	Sequence(Random)	Earliness	Tardiness	Makesspan	C.O.F
1	09-06-02-08-05-01-10-03-04-07	9810	30035	8730	0.9624
2	10-03-07-04-06-05-08-09-02-01	5245	22865	4835	0.6864
3	08-04-03-02-01-06-10-07-05-09	6875	27430	8810	0.8995
4	08-04-09-06-01-07-02-10-03-05	4065	24040	8915	0.7584
5	09-03-05-02-01-07-08-10-04-06	2745	48845	9110	0.8758
6	02-03-01-07-06-10-05-09-04-08	2160	56890	9850	0.9489
7	01-08-02-04-07-09-06-05-03-10	6815	24095	9870	0.9542
8	08-05-04-06-03-09-01-07-02-10	3970	35445	8255	0.7904
9	07-06-05-03-08-04-01-09-02-10	6145	44335	7320	0.7186
10	06-03-04-07-10-08-02-09-01-05	7385	31820	4360	0.8922
11	09-01-05-06-07-03-04-10-08-02	2670	52300	9865	0.9368
12	05-09-10-04-03-06-07-01-02-08	5045	36285	9725	0.9299
13	01-08-04-03-02-09-07-06-05-10	3340	36455	9930	0.8698
14	05-10-07-04-06-02-03-08-09-01	3140	38330	9115	0.8199
15	10-09-02-04-05-08-03-06-01-07	8475	32235	4195	0.741
16	06-03-05-04-10-08-07-01-09-02	8305	35455	4995	0.8104

17	04-09-01-08-07-10-06-05-03-02	9180	37440	9230	0.9632
18	05-10-09-04-06-02-01-07-08-03	5570	35575	7975	0.8478
19	02-06-01-07-08-09-03-10-05-04	6630	40975	6595	0.8592
20	05-02-09-07-03-06-08-04-10-01	1390	50230	8225	0.7869
21	04-02-05-08-06-09-01-03-10-07	6475	38465	7335	0.8866
22	04-01-07-03-06-10-08-05-09-02	4260	41130	9605	0.9058
23	03-09-05-01-04-06-10-07-02-08	3070	30150	7965	0.7057
24	07-10-05-03-09-02-04-08-01-06	4930	35860	7995	0.8396
25	10-01-09-02-07-05-06-08-04-03	3820	30790	7440	0.7248
26	07-10-06-04-05-08-03-02-01-09	2425	40740	8210	0.7495
27	01-08-09-06-10-05-03-04-02-07	535	27930	8645	0.6948
28	04-06-01-03-08-05-02-07-09-10	6350	26255	4010	0.6948
29	04-10-07-06-09-08-01-05-02-03	4735	37620	9745	0.9129
30	03-08-01-02-05-07-09-04-10-06	4285	38240	8675	0.8489

Additionally, a cumulative probability table has been formed. Cumulative probability has been formed by following the below mentioned procedures,

Expected value E = minimum value + (maximum – minimum value)*(R-1) / (N-1)

In the proposed problem, Minimum value is taken as 0.4 and maximum value is taken as 1.6.These minimum values and maximum values can be varied according to requirement. Cumulative probability set is then calculated by using the probability set values. Values calculated for N schedules have been tabulated in table 7.2

TABLE 7.2. CUMULATIVE PROBABILITY VALUES

Sl.n.o	Expected value (E)	Probability set (E/30)	Cumulative probability
1	0.4	0.0133	0.0133
2	0.4413	0.0147	0.028
3	0.4287	0.0160	0.044
4	0.5241	0.0174	0.0614
5	0.5655	0.0188	0.0802
6	0.6068	0.0202	0.1004
7	0.6482	0.0216	0.122
8	0.6896	0.0229	0.1449
9	0.7310	0.0243	0.1692
10	0.7724	0.0257	0.1949
11	0.8137	0.0271	0.2220
12	0.8551	0.0285	0.2505
13	0.8965	0.0299	0.2903
14	0.9379	0.0312	0.3115
15	0.9793	0.0326	0.3441
16	1.0206	0.0340	0.3781
17	1.0620	0.0354	0.4135
18	1.1034	0.0367	0.4502
19	1.1448	0.0381	0.4883
20	1.1862	0.0395	0.5278
21	0.0409	0.0409	0.5687
22	0.0422	0.0422	0.6109

23	0.0436	0.0436	0.6545
24	0.0450	0.0450	0.6995
25	0.0460	0.0460	0.7459
26	0.0478	0.0478	0.7937
27	0.0491	0.0491	0.8428
28	0.0505	0.0505	0.8933
29	0.0519	0.0519	0.9452
30	0.0533	0.0533	0.9985

7.5. OUTPUT OF REPRODUCTION

After computing cumulative probability values, each schedule is assigned a random number from range 0.000 to 0.999, by using random table. Rank has been selected for each schedule by comparing the random number for each schedule with cumulative probability values. N/2 pairs are then being selected for crossover in the order of selected rank. Random number assigned, selected rank for N schedules has been tabulated below.

TABLE 7.3. RANK BASED SELECTION

S. no	Sorted C.O.F	Rank	Probabability Of Selection	Cumulative Probability	Random Number	Rank Selected
1	0.9632	1	0.0133	0.0133	0.218	11
2	0.9624	2	0.0147	0.023	0.112	07
3	0.9542	3	0.0160	0.044	0.711	25
4	0.9489	4	0.0174	0.0614	0.655	24
5	0.9368	5	0.0188	0.0802	0.419	18
6	0.9299	6	0.0202	0.1004	0.354	16

7	0.9129	7	0.0216	0.122	0.174	10
8	0.9058	8	0.0229	0.1449	0.910	29
9	0.8995	9	0.0243	0.1692	0.076	5
10	0.8922	10	0.0257	0.1949	0.249	12
11	0.8866	11	0.0271	0.2222	0.129	8
12	0.8758	12	0.0285	0.2505	0.439	18
13	0.8698	13	0.0298	0.2803	0.380	17
14	0.8592	14	0.0312	0.3115	0.498	20
15	0.8478	15	0.0326	0.3441	0.134	8
16	0.8489	16	0.0340	0.3781	0.159	9
17	0.8396	17	0.0354	0.413517	0.966	6
18	0.8199	18	0.0367	0.4502	0.761	26
19	0.8104	19	0.0381	0.4883	0.850	28
20	0.7904	20	0.0395	0.5278	0.697	24
21	0.7869	21	0.0409	0.5687	0.579	22
22	0.7584	22	0.0422	0.6109	0.636	23
23	0.7495	23	0.0436	0.6545	0.416	18
24	0.741	24	0.0450	0.6995	0.035	3
25	0.7248	25	0.0464	0.7459	0.913	29
26	0.7186	26	0.0478	0.7937	0.512	20
27	0.7057	27	0.0491	0.8428	0.628	23
28	0.6948	28	0.0505	0.8933	0.752	26
29	0.6893	29	0.0519	0.9452	0.897	29
30	0.6864	30	0.0533	0.9985	0.232	12

In proposed problem, cross over probability (P_c) is varied from 0.5 to 0.9 and cross over site is varied from 0 to 9, depending on the value of cross over probability and cross over site , new schedules are generated.

Before cross over.

01-02-03-040-05-06-07-08-09-10

02-04-10-06-08-01-03-050-70-09

If the cross over site is 5

01-02-03-04-05|-06-07-08-09-10

02-04-10-06-08| -01-05-03-07-09

After cross over,

01-02-03-04-05-10-06-08-07-09

02-04-10-06-08-01-03-05-07-09

TABLE 7.4.CROSS OVER

S.no	Schedule after reproduction	Cross over site	Cross over yes/no	Schedule after cross over
1	01-08-02-04-07-09-06-05-03-10	03	Yes	01-08-02-10-09-04-05-03-06-07
2	10-09-02-04-05-08-03-06-01-07			10-09-02-01-08-04-07-06-05-03
3	08-04-09-06-01-07-02-10-03-05	07	NO	08-04-09-06-01-07-02-10-03-05
4	04-06-01-03-08-05-02-07-09-10			04-06-01-03-08-05-02-07-09-10
5	10-01-09-02-07-05-06-08-04-03	06	Yes	10-01-09-02-07-05-06-08-04-03
6	10-01-09-02-07-05-06-08-04-03			10-01-09-02-07-05-06-08-04-03
7	08-04-03-02-01-06-10-07-05-09	02	Yes	08-04-09-06-02-05-01-10-03-07
8	09-06-02-08-05-01-10-03-04-07			09-06-08-04-03-02-01-10-07-05
9	04-09-01-08-07-10-06-05-03-02	03	Yes	04-09-01-10-07-06-08-05-02-03
10	04-10-07-06-09-08-01-05-02-03			04-10-07-09-01-08-06-05-03-02
11	10-03-07-04-06-05-08-09-02-01	08	No	10-03-07-04-06-05-08-09-02-01

12	10-01-09-02-07-05-06-08-04-03			10-01-09-02-07-05-06-08-04-03
13	04-09-01-08-07-10-06-05-03-02	09	No	04-09-01-08-07-10-06-05-03-02
16	07-10-06-04-05-08-03-02-01-09			07-10-06-04-08-09-01-02-03-05
17	04-10-07-06-09-08-01-05-02-03	05	Yes	04-10-07-06-09-01-03-08-05-02
18	04-01-07-03-06-10-08-05-09-02			04-01-07-03-06-10-09-08-05-02
19	02-06-01-07-08-09-03-10-05-04	03	Yes	02-06-01-08-05-04-03-09-07-10
20	08-05-04-06-03-09-01-07-02-10			08-05-04-02-06-01-07-09-03-10
21	06-03-05-04-10-08-07-01-09-02	02	Yes	06-03-07-10-05-09-02-04-08-01
22	07-10-05-03-09-02-04-08-01-06			07-10-06-03-05-04-08-01-09-02
23	08-04-09-06-01-07-02-10-03-05	02	Yes	08-04-06-03-05-10-07-01-09-02
24	06-03-05-04-10-08-07-01-09-02			06-03-08-04-09-01-07-02-10-05
25	06-03-05-07-10-08-02-09-01-05	02	Yes	06-03-08-04-02-01-10-07-05-09
26	08-04-03-02-01-06-10-07-05-09			08-04-06-03-07-10-02-09-05
27	05-10-07-04-06-02-03-08-09-01	01	Yes	05-07-06-03-08-04-01-09-02-10
28	07-06-05-03-08-04-01-09-02-10			07-05-10-04-06-02-03-08-09-01
29	08-04-09-06-01-07-02-10-03-05	02	Yes	08-04-06-03-07-10-02-09-01-05
30	06-03-04-07-10-08-02-09-01-05			06-03-08-04-09-01-07-02-10-05

P-1656

7.6. METHOD OF MUTATION

Mutation is a random modification of a randomly selected string. Mutation is done with a mutation probability of 0.01 to 0.03.

For example,

In any two sequences any two positions are changed.

Before Mutation,

01-02-03-040-05-06-07-08-09-10

02-04-10-06-08-01-03-050-70-09

After Mutation,

01-03-02-040-05-06-07-08-09-10

02-04-06-10-08-01-03-050-70-09

TABLE 7.5 MUTATION

S.no	After Crossover	Mutation Site	After Mutation
1	01-08-02-10-09-04-05-03-06-07		01-08-02-10-09-04-05-03-06-07
2	10-09-02-01-08-04-07-06-05-03		10-09-02-01-08-04-07-06-05-03
3	08-04-09-06-01-07-02-10-03-05		08-04-09-06-01-07-02-10-03-05
4	04-06-01-03-08-05-02-07-09-10		04-06-01-03-08-05-02-07-09-10
5	10-01-09-02-07-05-06-08-04-03	02 & 03	10-09-01-02-07-05-06-08-04-03
6	10-01-09-02-07-05-06-08-04-03		10-01-09-02-07-05-06-08-04-03
7	08-04-09-06-02-05-01-10-03-07		08-04-09-06-02-05-01-10-03-07
8	09-06-08-04-03-02-01-10-07-05		09-06-08-04-03-02-01-10-07-05
9	04-09-01-10-07-06-08-05-02-03		04-09-01-10-07-06-08-05-02-03
10	04-10-07-09-01-08-06-05-03-02		04-10-07-09-01-08-06-05-03-02
11	10-03-07-04-06-05-08-09-02-01		10-03-07-04-06-05-08-09-02-01
12	10-01-09-02-07-05-06-08-04-03		10-01-09-02-07-05-06-08-04-03

13	04-09-01-08-07-10-06-05-03-02		04-09-01-08-07-10-06-05-03-02
14	07-10-06-04-05-08-03-02-01-09		07-10-06-04-05-08-03-02-01-09
15	08-04-09-06-07-10-05-03-02-01	05 & 06	08-04-09-06-10-07-05-03-02-01
16	07-10-06-04-08-09-01-02-03-05		07-10-06-04-08-09-01-02-03-05
17	04-10-07-06-09-01-03-08-05-02		04-10-07-06-09-01-03-08-05-02
18	04-01-07-03-06-10-09-08-05-02		04-01-07-03-06-10-09-08-05-02
19	02-06-01-08-05-04-03-09-07-10		02-06-01-08-05-04-03-09-07-10
20	08-05-04-02-06-01-07-09-03-10		08-05-04-02-06-01-07-09-03-10
21	06-03-07-10-05-09-02-04-08-01		06-03-07-10-05-09-02-04-08-01
22	07-10-06-03-05-04-08-01-09-02		07-10-06-03-05-04-08-01-09-02
23	08-04-06-03-05-10-07-01-09-02		08-04-06-03-05-10-07-01-09-02
24	06-03-08-04-09-01-07-02-10-05		06-03-08-04-09-01-07-02-10-05
25	06-03-08-04-02-01-10-07-05-09		06-03-08-04-02-01-10-07-05-09
26	08-04-06-03-07-10-02-09-05		08-04-06-03-07-10-02-09-05
27	05-07-06-03-08-04-01-09-02-10		05-07-06-03-08-04-01-09-02-10
28	07-05-10-04-06-02-03-08-09-01		07-05-10-04-06-02-03-08-09-01
29	08-04-06-03-07-10-02-09-01-05		08-04-06-03-07-10-02-09-01-05
30	06-03-08-04-09-01-07-02-10-05	02 & 08	06-02-08-04-09-01-07-03-10-05

7.7. OUTPUT AFTER THE FIRST ITERATION

The best string obtained after performing 1st generation is the chromosome number 30, which has the minimum, C.O.F value of 0.720979. The corresponding earliness is 5245, tardiness is22865 and makespan is 4835. This completes one generation of the GA and the best value is stored. All the strings available at the end of the first iteration will be treated as parents for the 2nd iteration. This procedure is repeated for the number of iterations as given by the user.

TABLE 7.6. OUTPUT AFTER THE FIRST GENERATION

S.no	Schedule after crossover and mutation	Earliness	Tardiness	Makespan	C.o.f
1	01-08-02-10-09-04-05-03-06-07	6815	24095	9870	1.12641
2	10-09-02-01-08-04-07-06-05-03	8475	32235	4195	0.89432
3	08-04-09-06-01-07-02-10-03-05	4065	24040	8915	0.94227
4	04-06-01-03-08-05-02-07-09-10	6350	26255	4010	0.74239
5	10-09-01-02-07-05-06-08-04-03	3305	32860	7485	0.88346
6	10-01-09-02-07-05-06-08-04-03	3820	30790	7440	0.88749
7	08-04-09-06-02-05-01-10-03-07	6875	27430	8810	1.08588
8	09-06-08-04-03-02-01-10-07-05	9810	30035	8730	1.23064
9	04-09-01-10-07-06-08-05-02-03	9180	37440	9230	1.29189
10	04-10-07-09-01-08-06-05-03-02	4735	37620	9745	1.12980
11	10-03-07-04-06-05-08-09-02-01	5245	22865	4835	0.72097
12	10-01-09-02-07-05-06-08-04-03	3820	30790	7440	0.88749
13	04-09-01-08-07-10-06-05-03-02	9180	37440	9230	1.29189
14	07-10-06-04-05-08-03-02-01-09	2425	40470	8210	0.94979
15	08-04-09-06-10-07-05-03-02-01	3740	26225	9800	1.00200
16	07-10-06-04-08-09-01-02-03-05	2425	40740	8210	0.95186
17	04-10-07-06-09-01-03-08-05-02	4725	37620	9745	1.12936
18	04-01-07-03-06-10-09-08-05-02	4260	41130	9605	1.12657
19	02-06-01-08-05-04-03-09-07-10	6630	40975	6595	1.03518

20	08-05-04-02-06-01-07-09-03-10	3970	35445	8255	0.98264
21	06-03-07-10-05-09-02-04-08-01	8305	35445	4995	0.96325
22	07-10-06-03-05-04-08-01-09-02	4930	35860	7995	1.01148
23	08-04-06-03-05-10-07-01-09-02	4065	24840	8915	0.9484
24	06-03-08-04-09-01-07-02-10-05	8305	35455	4995	0.96335
25	06-03-08-04-02-01-10-07-05-09	7385	31820	4360	0.85356
26	08-04-06-03-07-10-02-09-05	6875	27230	8810	1.08435
27	05-07-06-03-08-04-01-09-02-10	3140	38330	9115	1.02375
28	07-05-10-04-06-02-03-08-09-01	6145	44335	7320	1.08645
29	08-04-06-03-07-10-02-09-01-05	4065	24040	8915	0.94227
30	06-02-08-04-09-01-07-03-10-05	6275	25100	5560	0.83073

7.8. ALGORITHM:

Step 1: Start the program.

Step 2: Get the input values of Crossover rate, Mutation rate, Number of iterations, Weightages for Makespan, Earliness, Tardiness.

Step 3: Input the values like quantity, time of operation in each stage for each job.

Step 4: A sequence is generated at random and for that sequence the jobs are assigned taking into consideration the number of machines in each stage and the order in which they are present as follows,

a) Assign the jobs as per schedule to the machines.
b) If all the machines are occupied with jobs the next job can be assigned to a machine only when any one of the machine becomes idle after completion of previous operation.
c) Jobs should be assigned to machines in second stage as soon as they complete their operation in stage 1.

Step 4: Calculate the values of maximum makespan, earliness, tadiness for each sequence for all thirty sequences .

Step 5: Calculate the value of C.O.F

Step 6: Rank all the thirty sequences in the decending order of C.O.F values .

Step 7: Sequences with higher C.O.F values are removed by comparing them with cumulative probability values and those sequences are removed and better sequences are replace the removed ones.

Step 8: A random crossover site is selected and sequence pairs are arranged in the order of the other sequence in the pair after the crossover site.

Step 9: Two random numbers are generated for each sequence and the corresponding strings are swapped.This completes iteration.

Step 10: These thirty sequences are taken as input and all these above operations are repeated for these sequences forming a loop until the number of iterations preferred is over. Best C.O.F values of iteration are taken and plotted on a graph

7.9. VB PROGRAM

```
Dim j As String
Dim JobEd As Integer, ed As Boolean
''Dim seq1, seq2 As String
Dim mutAr(1 To 30) As Integer, mutArNo(1 To 30) As Integer
Dim mA(1 To 30) As String
Dim CoP(1 To 30) As Integer
Dim ar() As Long  ' for mutation sorting
Dim mak(0 To 4) As Boolean ' for checking in case the minitues are same
Dim TotMakeSpan As Long, GMkSp
Dim COPRank(30) As Integer
Dim IterationNo As Integer
Dim EndProcess As Boolean
Dim sch1() As Integer
```

```
Dim chrtStatic As Integer
Dim chartAr(1000) As Double

Dim aa As Integer
Dim rndar() As Integer 'December 7 correction

Private Sub CalculateCOF()
Dim item1(1 To 30) As Double, item2(1 To 30) As Double, item3(1 To 30) As
Double
Dim MakeSpanTotal As Long, EarlinessTotal As Long, TardinessTotal As Long
Dim COFValue As Double

Static counter As Integer
counter = counter + 1
If counter > 30 Then counter = 1
MakeSpanTotal = CLng(MakeSpanTotal) +
CLng(lstMakespan.ListItems(counter).SubItems(2))
EarlinessTotal = CLng(EarlinessTotal) +
CLng(lstMakespan.ListItems(counter).SubItems(3))
TardinessTotal = CLng(EarlinessTotal) +
CLng(lstMakespan.ListItems(counter).SubItems(4))

If IterationNo = 1 Then
   AverageMakeSpan = MakeSpanTotal / 30
   AverageEarliness = EarlinessTotal / 30
   AverageTardiness = TardinessTotal / 30
End If

Dim tmp1 As Long, tmp2 As Long, tmp3 As Long

tmp1 = CLng(lstMakespan.ListItems(counter).SubItems(2))
tmp2 = CLng(lstMakespan.ListItems(counter).SubItems(3))
tmp3 = CLng(lstMakespan.ListItems(counter).SubItems(4))
```

```
item1(counter) = CDbl(var1 * (tmp1 / AverageMakeSpan))
item2(counter) = CDbl(var2 * (tmp2 / AverageEarliness))
item3(counter) = CDbl(var3 * (tmp3 / AverageTardiness))
COFValue = item1(counter) + item2(counter) + item3(counter)
lstMakespan.ListItems(counter).SubItems(5) = Format(COFValue, "0.00000000")

End Sub

Private Sub CalculateMakeSpanStage1()

ClearStage1and2ListBoxes
Stage1
Allot1
Stage2
Allot2
SortList
EarlinessTardiness
CalculateCOF
RankCOF

End Sub

Private Sub RankCOF()
Static counter As Integer
counter = counter + 1
If counter > 30 Then counter = 1
   listCOFRank.AddItem     Format(lstMakespan.ListItems(counter).SubItems(5),
"0.00000000") ' for sorting and ranking

   If counter = 30 Then

      Dim rnk As Integer, i As Integer, q As Integer
      rnk = 1
```

```
    For i = 30 To 1 Step -1
      For q = 30 To 1 Step -1
        If listCOFRank.List(i - 1) = lstMakespan.ListItems(q).SubItems(5) Then
          lstMakespan.ListItems(q).SubItems(6) = Format(rnk, "00")
          rnk = rnk + 1
          Exit For
        End If
      Next q
    Next i

    CheckBlanks
    Reproduction
    CrossOver1
    CrossOver2
    Swap
    WriteToFile
  End If

End Sub

Private Sub CheckBlanks()
Dim i As Integer
For i = 1 To 30
  If Trim(lstMakespan.ListItems(i).SubItems(6)) = "" Then
    tmp = lstMakespan.ListItems(i).SubItems(5)
    pos = i

    For k = 1 To 30
      If lstMakespan.ListItems(k).SubItems(5) = tmp And
lstMakespan.ListItems(k).SubItems(6) <> "" Then
        lstMakespan.ListItems(pos).SubItems(6) =
Val(lstMakespan.ListItems(k).SubItems(6)) + 1
        Exit For
      End If
```

```
        Next k
    End If
Next i

  End Sub

Private Sub Reproduction()
Dim X As Integer, Y As Integer
Dim l, m As ListItem, i As Integer

For i = 1 To 30
Randomize
    Set m = lstReproduction.ListItems.Add(, , Format(i, "00"))
    m.SubItems(1) = ""
    m.SubItems(2) = Format(CumProb(i - 1), "0.0000")
    m.SubItems(3) = Format((Rnd()) + 0.0001, "0.0000")
    m.SubItems(4) = ""
Next i
For X = 1 To 30
    For Y = 1 To 30
        If Val(lstMakespan.ListItems(Y).SubItems(6)) = X Then
         lstReproduction.ListItems(X).SubItems(1)                                    =
lstMakespan.ListItems(Y).SubItems(1)
        Exit For
        End If
    Next Y
Next X

Dim pos As Integer, counter As Integer
Dim tmp As Double, tmp2 As Double
For X = 1 To 30
    For Y = 1 To 30
        If    CDbl(lstReproduction.ListItems(Y).SubItems(3)    >=    0.9985)    Then
lstReproduction.ListItems(Y).SubItems(3) = 0.9985
```

```
    If              CDbl(lstReproduction.ListItems(Y).SubItems(2))              >=
CDbl(lstReproduction.ListItems(X).SubItems(3)) Then
        counter = counter + 1
        If counter = 1 Then
          tmp       =       CDbl(lstReproduction.ListItems(Y).SubItems(2))       -
CDbl(lstReproduction.ListItems(X).SubItems(3))
          pos = Y
        End If
          If counter > 1 Then
          tmp2      =       CDbl(lstReproduction.ListItems(Y).SubItems(2))       -
CDbl(lstReproduction.ListItems(X).SubItems(3))
            If tmp2 < tmp Then
              tmp = tmp2

              pos = Y
            End If
         End If
      End If

    Next Y
    If pos > 0 Then
    lstReproduction.ListItems(X).SubItems(4)                                      =
lstReproduction.ListItems(pos).SubItems(1)
    "lstReproduction.ListItems(x).SubItems(4) = pos '
    tmp = 0
    tmp2 = 0
    pos = 0
    counter = 0
    End If
Next X

End Sub
```

```
Private Sub CrossOver1()
   Randomize
   Dim m As ListItem, i As Integer
   lstCrossOver.ListItems.Clear

   For i = 1 To 30
      mutAr(i) = 0
      mutArNo(i) = 0
      mA(i) = ""
      CoP(i) = 0
   Next i

   For i = 1 To 30
      Randomize
      Set m = lstCrossOver.ListItems.Add(, , Format(i, "00"))
      m.SubItems(1)   =   lstReproduction.ListItems(i).SubItems(4)   '   Random
sequence

      If i Mod 2 = 1 Then
         m.SubItems(3) = Format(Int((Rnd() * 9) + 1), "00")
         m.SubItems(2) = IIf(m.SubItems(3) <= CORATE, "Yes", "No")

End If

         mA(i) = m.SubItems(1)
         CoP(i) = Val(m.SubItems(3))

      If m.SubItems(2) = "Yes" Then
         mutAr(i) = i
         List11.AddItem m.SubItems(2)
         List12.AddItem mutAr(i)
         counter = counter + 1
```

```
End If
        If m.SubItems(2) = "No" Then mutArNo(i) = i
    Next i

End Sub


Private Sub CrossOver2()
Dim spAr1() As String ' Array for splitting
Dim spAr2() As String

ReDim t1(1 To No_of_Jobs) As Integer
ReDim t2(1 To No_of_Jobs) As Integer

ReDim X(1 To No_of_Jobs) As Integer
ReDim Y(1 To No_of_Jobs) As Integer

ReDim tmp(1 To No_of_Jobs) As Integer
Dim pt As Integer, i As Integer, r As Integer
Dim s As String

List107.Clear
List108.Clear
List109.Clear
List110.Clear
List111.Clear
List112.Clear
List113.Clear

For i = 1 To No_of_Jobs
    t1(i) = 0
    t2(i) = 0
    X(i) = 0
    Y(i) = 0
```

```
    tmp(i) = 0
    s = ""
Next i

For i = 1 To UBound(mutAr)
  If mutAr(i) <> 0 Then
     List107.AddItem mA(i)
     List107.AddItem mA(i + 1)
     List108.AddItem CoP(i)

 End If
Next i


For Z = 1 To List108.ListCount

   For i = 1 To No_of_Jobs
   t1(i) = 0
   t2(i) = 0
   X(i) = 0
   Y(i) = 0
   tmp(i) = 0
   s = ""
   Next i

   List109.Clear
   List110.Clear
   List111.Clear
   List112.Clear

   pt = Val(List108.List(Z - 1))

   spAr1 = Split(List107.List(((Z - 1) * 2)), "-")
   spAr2 = Split(List107.List(((Z - 1) * 2) + 1), "-")
```

```
For i = 0 To UBound(spAr1)
t1(i + 1) = Val(spAr1(i))
Next i
For i = 0 To UBound(spAr2)
t2(i + 1) = Val(spAr2(i))
Next i

For i = 1 To pt
List109.AddItem Str(Format$(t1(i))) 'first of the pair
List111.AddItem Str(Format$(t2(i))) 'second
Next i

For i = pt + 1 To No_of_Jobs ' for the first item in the pair
   For w = 1 To No_of_Jobs
      If (t1(i) = t2(w)) Then
         tmp(w) = t1(i)
         Exit For
      End If
   Next w
Next i

For i = 1 To No_of_Jobs
  If Val(tmp(i)) <> 0 Then List110.AddItem Str(Format$(tmp(i)))
Next i

For i = pt + 1 To No_of_Jobs
   For w = 1 To No_of_Jobs
      If (t2(i) = t1(w)) Then
         X(w) = t2(i)
         Exit For
      End If
   Next w
Next i
For i = 1 To No_of_Jobs
```

```
  If Val(X(i)) <> 0 Then List112.AddItem Str(Format$(X(i), "00"))
Next i

s = ""
For i = 0 To List109.ListCount - 1
  s = s & Str(Format$(List109.List(i), "00"))
Next i
For i = 0 To List110.ListCount - 1
  s = s & Str(Format$(List110.List(i), "00"))
Next i
List113.AddItem Format$(Replace(s, " ", "-0"), "00")
s = ""
For i = 0 To List111.ListCount - 1
  s = s & Str(Format$(List111.List(i), "00"))
Next i
For i = 0 To List112.ListCount - 1
  s = s & Str(Format$(List112.List(i), "00"))
Next i
List113.AddItem Format$(Replace(s, " ", "-0"), "00")
s = ""
Next Z

For i = 1 To 30 Step 2
  If mutAr(i) <> 0 Then
    List114.AddItem Replace(List113.List(c), "-010", "-10")
    List114.AddItem Replace(List113.List(c + 1), "-010", "-10")
    c = c + 2
  ElseIf mutArNo(i) <> 0 Then
    List114.AddItem lstCrossOver.ListItems(mutArNo(i)).SubItems(1)
    List114.AddItem lstCrossOver.ListItems(mutArNo(i) + 1).SubItems(1)
  End If
Next i
For i = 1 To 30
```

```
  lstCrossOver.ListItems(i).SubItems(4) = IIf(Left(List114.List(i - 1), 1) = "-",
Mid$(List114.List(i - 1), 2), List114.List(i - 1))
Next i


End Sub

Private Sub Swap()
Dim i As Integer
Dim swap1 As String, swap2 As String
Dim splitar() As String

Randomize
r1 = Int(Rnd() * 9) + 1
r2 = Int(Rnd() * 9) + 11
r3 = Int(Rnd() * 9) + 21
List114.Clear
For i = 1 To 30
List114.List(i - 1) = lstCrossOver.ListItems(i).SubItems(4)
Next i

For i = 0 To List114.ListCount - 1
  If i = r1 Or i = r2 Or i = r3 Then
      splitar = Split(List114.List(i), "-")
      Randomize
      r = Int(Rnd() * No_of_Jobs - 1)
      If r <= 0 Then r = 1: If r >= 8 Then r = 8
      swap1 = splitar(r)
      swap2 = splitar(r + 1)

      For k = 0 To UBound(splitar)
         If k = r - 1 Then
         splitar(r) = swap2
         splitar(r + 1) = swap1
```

```
            End If
            s = s + Format$(splitar(k), "00") + "-"
        Next k
        lstSwap.AddItem s
        s = ""
    Else
        lstSwap.AddItem lstCrossOver.ListItems(i + 1).SubItems(1)
    End If
Next i

Dim lw As ListItem
Dim lft As String
lstMutation.ListItems.Clear
    For i = 1 To 30
        Set lw = lstMutation.ListItems.Add(, , Format(i, "00"))
        lw.SubItems(1) = lstCrossOver.ListItems(i).SubItems(4)

        If Left$(lstSwap.List(i - 1), 1) = "-" Then
        lft = Mid$(lstSwap.List(i - 1), 2)
        Else
        lft = lstSwap.List(i - 1)
        End If
        If Right$(lft, 1) = "-" Then
        lft = Mid$(lft, 1, Len(lft) - 1)
        Else
        lft = lft
        End If
        lw.SubItems(2) = lft
    Next i
    lstMutation.ListItems(r1 + 1).ForeColor = vbRed
    lstMutation.ListItems(r2 + 1).ForeColor = vbRed
    lstMutation.ListItems(r3 + 1).ForeColor = vbRed

End Sub
```

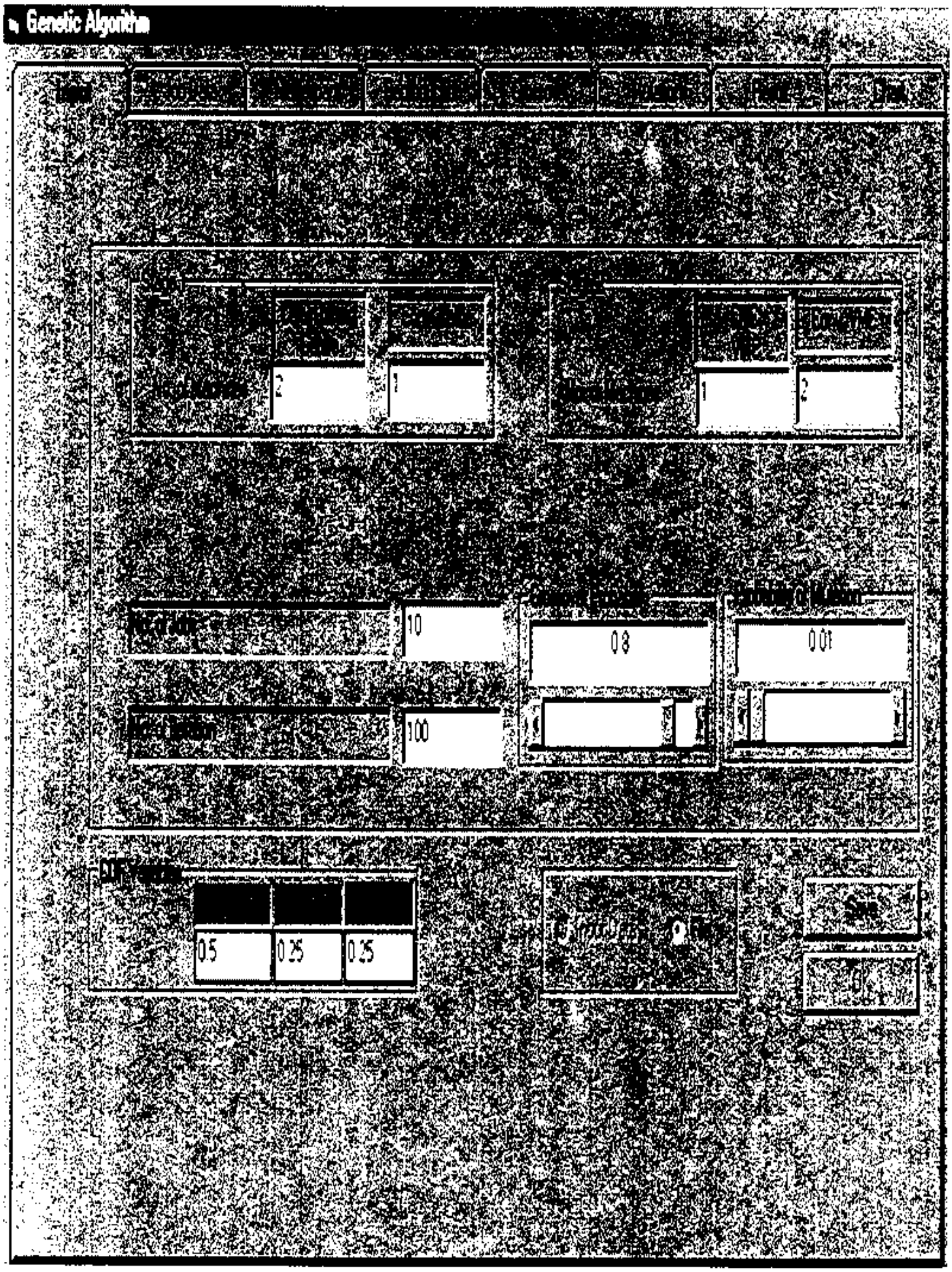


FIGURE 7.1 INPUT FORM OF THE VB PROGRAM

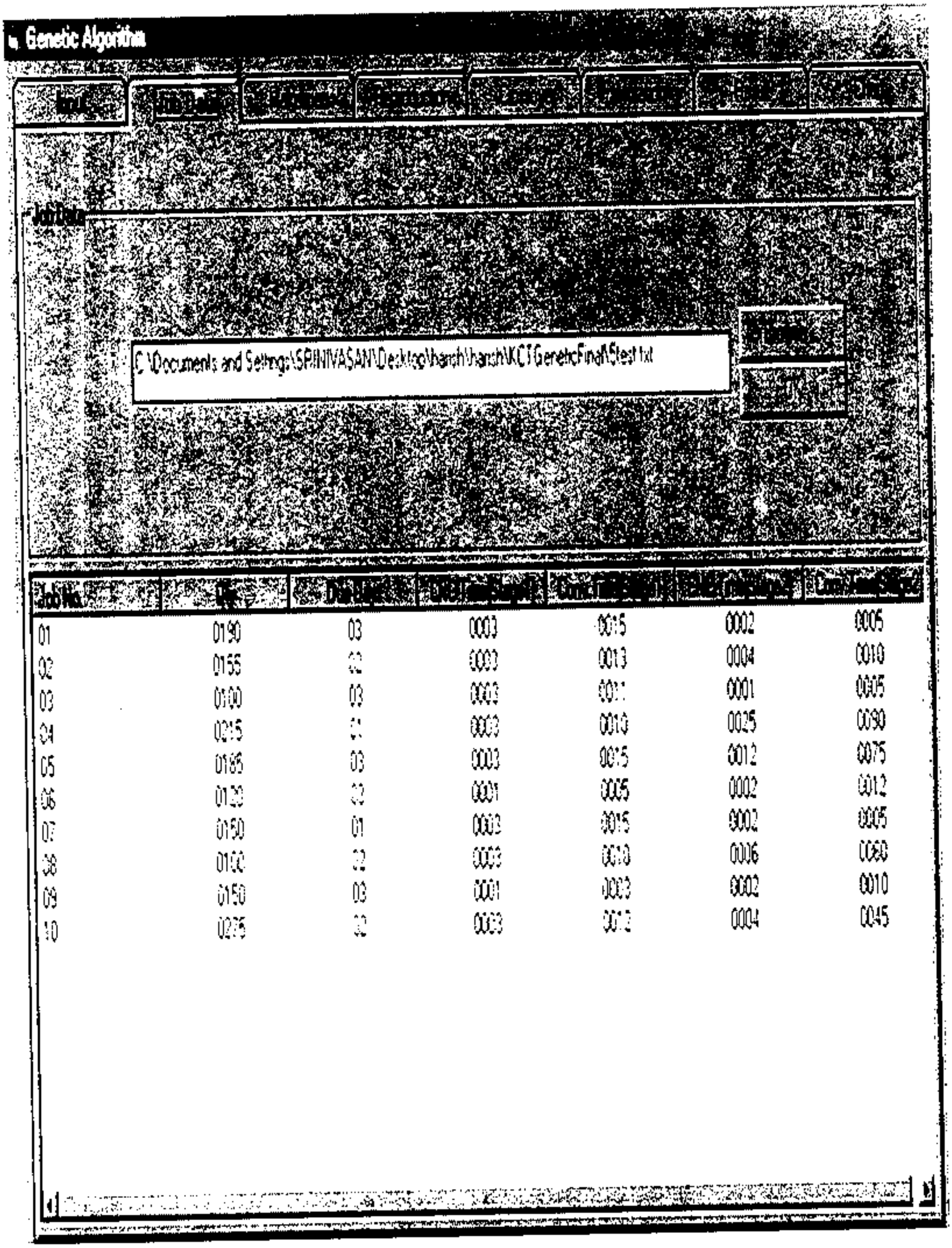
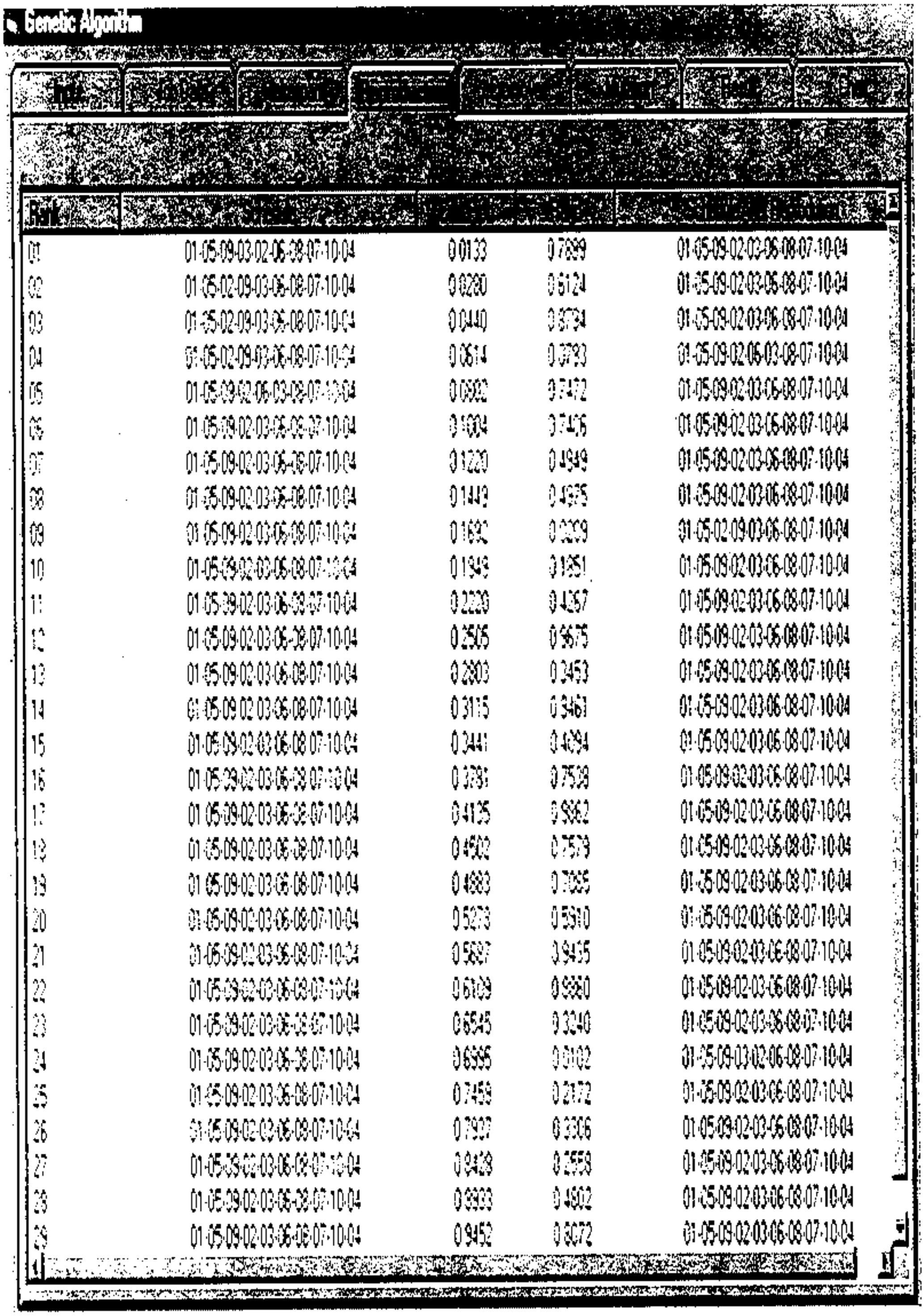


FIGURE 7.2 JOB DATA FORM OF THE VB PROGRAM

Genetic Algorithm

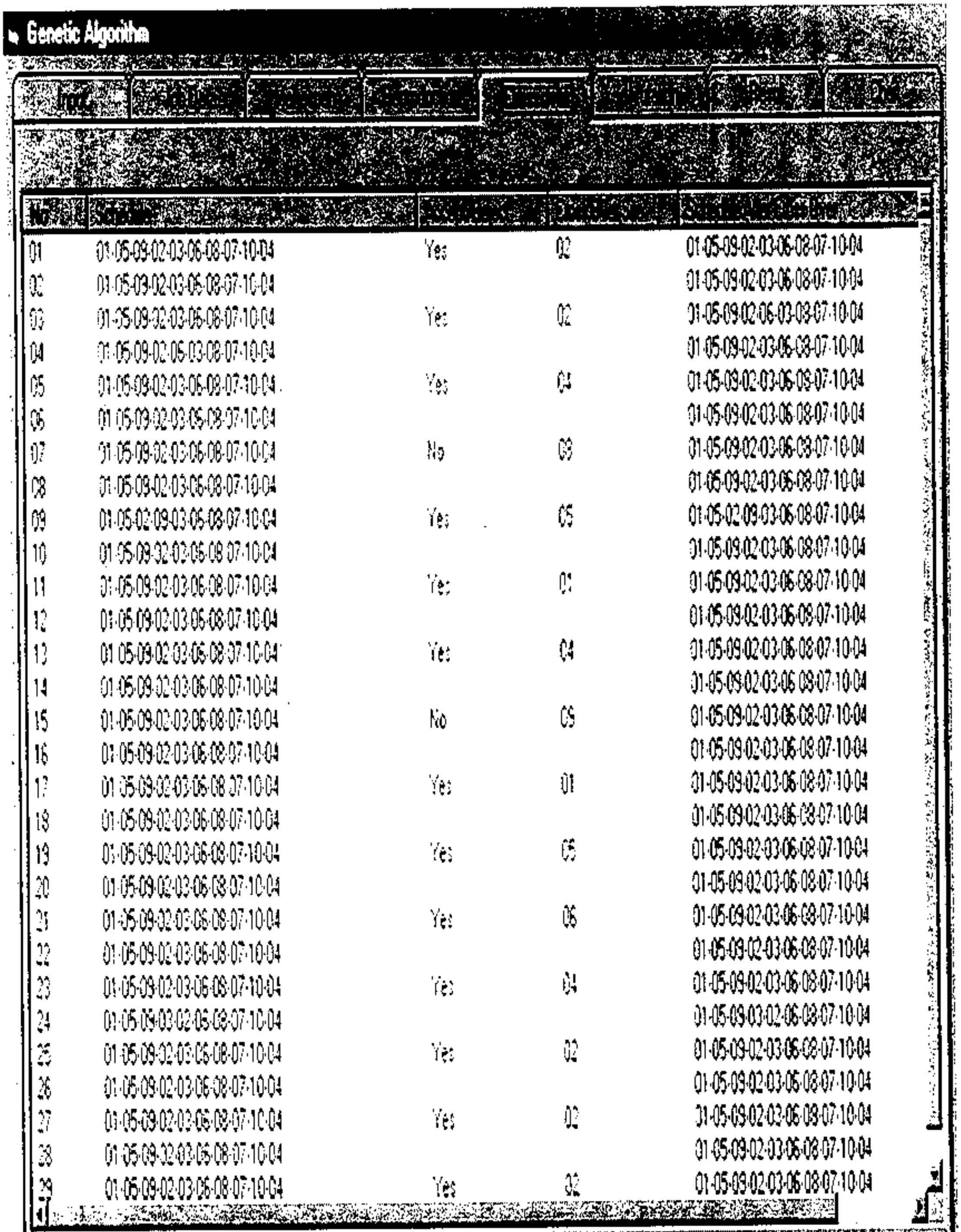
No	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	Rank
01	01-05-09-02-03-06-08-07-10-04	0006405	0001105	0029510	0.97949483	30
02	01-05-09-02-03-06-08-07-10-04	0006405	0001105	0029510	0.97949483	29
03	01-05-09-02-03-06-08-07-10-04	0006405	0001105	0029510	0.97949483	28
04	01-05-02-09-03-06-08-07-10-04	0006710	0001155	0026845	0.99068472	04
05	01-05-02-09-03-06-08-07-10-04	0006710	0001155	0026845	0.99068472	03
06	01-05-09-02-03-06-08-07-10-04	0006405	0001105	0029510	0.97949483	27
07	01-05-09-02-03-06-08-07-10-04	0006405	0001105	0029510	0.97949483	26
08	01-05-09-02-03-06-08-07-10-04	0006405	0001105	0029510	0.97949483	25
09	01-05-09-02-03-06-08-07-10-04	0006405	0001105	0029510	0.97949483	24
10	01-05-09-02-03-06-08-07-10-04	0006405	0001105	0029510	0.97949483	23
11	01-05-09-02-03-06-08-07-10-04	0006405	0001105	0029510	0.97949483	22
12	01-05-09-02-03-06-08-07-10-04	0006405	0001105	0029510	0.97949483	21
13	01-05-09-02-06-03-08-07-10-04	0006405	0001120	0029480	0.98233157	05
14	01-05-09-02-03-06-08-07-10-04	0006405	0001105	0029510	0.97949483	20
15	01-05-09-02-03-06-08-07-10-04	0006405	0001105	0029510	0.97949483	19
16	01-05-09-02-03-06-08-07-10-04	0006405	0001105	0029510	0.97949483	18
17	01-05-09-02-03-06-08-07-10-04	0006405	0001105	0029510	0.97949483	17
18	01-05-02-09-03-06-08-07-10-04	0006710	0001155	0026845	0.99068472	02
19	01-05-09-02-03-06-08-07-10-04	0006405	0001105	0029510	0.97949483	16
20	01-05-09-02-03-06-08-07-10-04	0006405	0001105	0029510	0.97949483	15
21	01-05-09-02-03-06-08-07-10-04	0006405	0001105	0029510	0.97949483	14
22	01-05-09-02-03-06-08-07-10-04	0006405	0001105	0029510	0.97949483	13
23	01-05-09-02-03-06-08-07-10-04	0006405	0001105	0029510	0.97949483	12
24	01-05-09-03-02-06-08-07-10-04	0006020	0004140	0026630	1.55544537	01
25	01-05-09-02-03-06-08-07-10-04	0006405	0001105	0029510	0.97949483	11
26	01-05-09-02-03-06-08-07-10-04	0006405	0001105	0029510	0.97949483	10
27	01-05-09-02-03-06-08-07-10-04	0006405	0001105	0029510	0.97949483	09
28	01-05-09-02-03-06-08-07-10-04	0006405	0001105	0029510	0.97949483	08
29	01-05-09-02-03-06-08-07-10-04	0006405	0001105	0029510	0.97949483	07

FIGURE 7.3 MAKESPAN, EARLINESS AND TARDINESS CALCULATED IN THE VB PROGRAM



Rank	[illegible]	[illegible]	[illegible]	[illegible]
01	01-05-09-03-02-06-08-07-10-04	0.0133	0.7699	01-05-09-02-03-06-08-07-10-04
02	01-05-02-09-03-06-08-07-10-04	0.0280	0.6124	01-05-09-02-03-06-08-07-10-04
03	01-05-02-09-03-06-08-07-10-04	0.0440	0.3734	01-05-09-02-03-06-08-07-10-04
04	01-05-02-09-03-06-08-07-10-04	0.0614	0.3793	01-05-09-02-06-03-08-07-10-04
05	01-05-09-02-06-03-08-07-10-04	0.0922	0.7472	01-05-09-02-03-06-08-07-10-04
06	01-05-09-02-03-06-08-07-10-04	0.1004	0.7406	01-05-09-02-03-06-08-07-10-04
07	01-05-09-02-03-06-08-07-10-04	0.1220	0.4949	01-05-09-02-03-06-08-07-10-04
08	01-05-09-02-03-06-08-07-10-04	0.1449	0.4975	01-05-09-02-03-06-08-07-10-04
09	01-05-09-02-03-06-08-07-10-04	0.1692	0.2209	01-05-02-09-03-06-08-07-10-04
10	01-05-09-02-03-06-08-07-10-04	0.1949	0.1951	01-05-09-02-03-06-08-07-10-04
11	01-05-09-02-03-06-08-07-10-04	0.2229	0.4257	01-05-09-02-03-06-08-07-10-04
12	01-05-09-02-03-06-08-07-10-04	0.2505	0.5675	01-05-09-02-03-06-08-07-10-04
13	01-05-09-02-03-06-08-07-10-04	0.2803	0.3453	01-05-09-02-03-06-08-07-10-04
14	01-05-09-02-03-06-08-07-10-04	0.3115	0.3461	01-05-09-02-03-06-08-07-10-04
15	01-05-09-02-03-06-08-07-10-04	0.3441	0.4094	01-05-09-02-03-06-08-07-10-04
16	01-05-09-02-03-06-08-07-10-04	0.3791	0.7529	01-05-09-02-03-06-08-07-10-04
17	01-05-09-02-03-06-08-07-10-04	0.4125	0.9862	01-05-09-02-03-06-08-07-10-04
18	01-05-09-02-03-06-08-07-10-04	0.4502	0.7579	01-05-09-02-03-06-08-07-10-04
19	01-05-09-02-03-06-08-07-10-04	0.4883	0.7055	01-05-09-02-03-06-08-07-10-04
20	01-05-09-02-03-06-08-07-10-04	0.5273	0.5510	01-05-09-02-03-06-08-07-10-04
21	01-05-09-02-03-06-08-07-10-04	0.5697	0.9435	01-05-09-02-03-06-08-07-10-04
22	01-05-09-02-03-06-08-07-10-04	0.6109	0.9920	01-05-09-02-03-06-08-07-10-04
23	01-05-09-02-03-06-08-07-10-04	0.6545	0.3240	01-05-09-02-03-06-08-07-10-04
24	01-05-09-02-03-06-08-07-10-04	0.6995	0.0102	01-05-09-03-02-06-08-07-10-04
25	01-05-09-02-03-06-08-07-10-04	0.7459	0.2172	01-05-09-02-03-06-08-07-10-04
26	01-05-09-02-03-06-08-07-10-04	0.7907	0.3306	01-05-09-02-03-06-08-07-10-04
27	01-05-09-02-03-06-08-07-10-04	0.8429	0.2553	01-05-09-02-03-06-08-07-10-04
28	01-05-09-02-03-06-08-07-10-04	0.8903	0.4802	01-05-09-02-03-06-08-07-10-04
29	01-05-09-02-03-06-08-07-10-04	0.9452	0.8072	01-05-09-02-03-06-08-07-10-04

FIGURE 7.4 REPRODUCTION PROCESS PERFORMED IN THE VB PROGRAM



No	[illegible]	[illegible]	[illegible]	[illegible]
01	01-05-09-02-03-06-08-07-10-04	Yes	02	01-05-09-02-03-06-08-07-10-04
02	01-05-09-02-03-06-08-07-10-04			01-05-09-02-03-06-08-07-10-04
03	01-05-09-02-03-06-08-07-10-04	Yes	02	01-05-09-02-06-03-08-07-10-04
04	01-05-09-02-06-03-08-07-10-04			01-05-09-02-03-06-08-07-10-04
05	01-05-09-02-03-06-08-07-10-04	Yes	04	01-05-09-02-03-06-08-07-10-04
06	01-05-09-02-03-06-08-07-10-04			01-05-09-02-03-06-08-07-10-04
07	01-05-09-02-03-06-08-07-10-04	No	09	01-05-09-02-03-06-08-07-10-04
08	01-05-09-02-03-06-08-07-10-04			01-05-09-02-03-06-08-07-10-04
09	01-05-02-09-03-06-08-07-10-04	Yes	05	01-05-02-09-03-06-08-07-10-04
10	01-05-09-02-03-06-08-07-10-04			01-05-09-02-03-06-08-07-10-04
11	01-05-09-02-03-06-08-07-10-04	Yes	01	01-05-09-02-03-06-08-07-10-04
12	01-05-09-02-03-06-08-07-10-04			01-05-09-02-03-06-08-07-10-04
13	01-05-09-02-03-06-08-07-10-04	Yes	04	01-05-09-02-03-06-08-07-10-04
14	01-05-09-02-03-06-08-07-10-04			01-05-09-02-03-06-08-07-10-04
15	01-05-09-02-03-06-08-07-10-04	No	09	01-05-09-02-03-06-08-07-10-04
16	01-05-09-02-03-06-08-07-10-04			01-05-09-02-03-06-08-07-10-04
17	01-05-09-02-03-06-08-07-10-04	Yes	01	01-05-09-02-03-06-08-07-10-04
18	01-05-09-02-03-06-08-07-10-04			01-05-09-02-03-06-08-07-10-04
19	01-05-09-02-03-06-08-07-10-04	Yes	05	01-05-09-02-03-06-08-07-10-04
20	01-05-09-02-03-06-08-07-10-04			01-05-09-02-03-06-08-07-10-04
21	01-05-09-02-03-06-08-07-10-04	Yes	06	01-05-09-02-03-06-08-07-10-04
22	01-05-09-02-03-06-08-07-10-04			01-05-09-02-03-06-08-07-10-04
23	01-05-09-02-03-06-08-07-10-04	Yes	04	01-05-09-02-03-06-08-07-10-04
24	01-05-09-03-02-06-08-07-10-04			01-05-09-03-02-06-08-07-10-04
25	01-05-09-02-03-06-08-07-10-04	Yes	02	01-05-09-02-03-06-08-07-10-04
26	01-05-09-02-03-06-08-07-10-04			01-05-09-02-03-06-08-07-10-04
27	01-05-09-02-03-06-08-07-10-04	Yes	02	01-05-09-02-03-06-08-07-10-04
28	01-05-09-02-03-06-08-07-10-04			01-05-09-02-03-06-08-07-10-04
29	01-05-09-02-03-06-08-07-10-04	Yes	02	01-05-09-02-03-06-08-07-10-04

FIGURE 7.5 CROSSOVER OPERATIONS IN THE VB PROGRAM

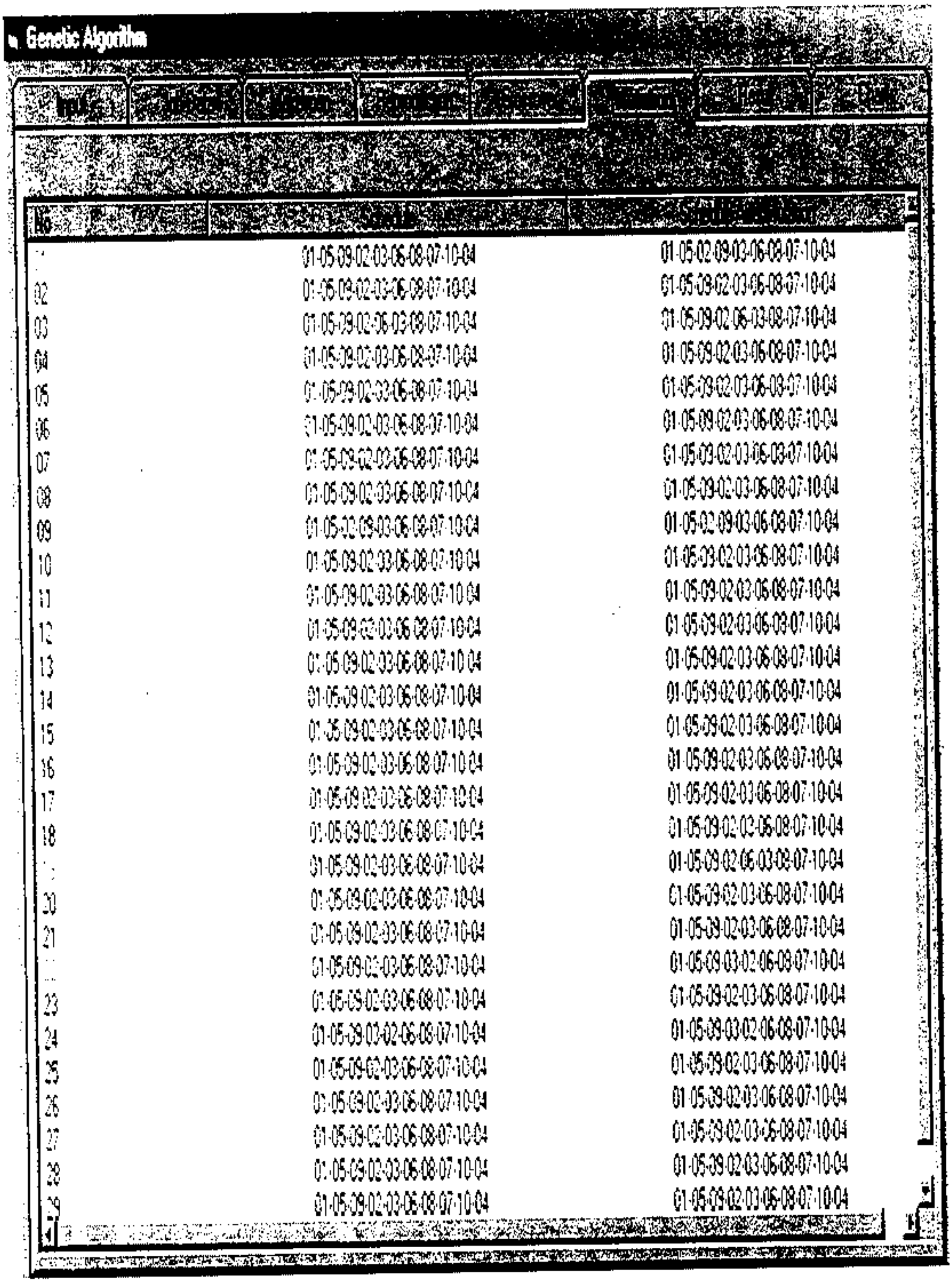
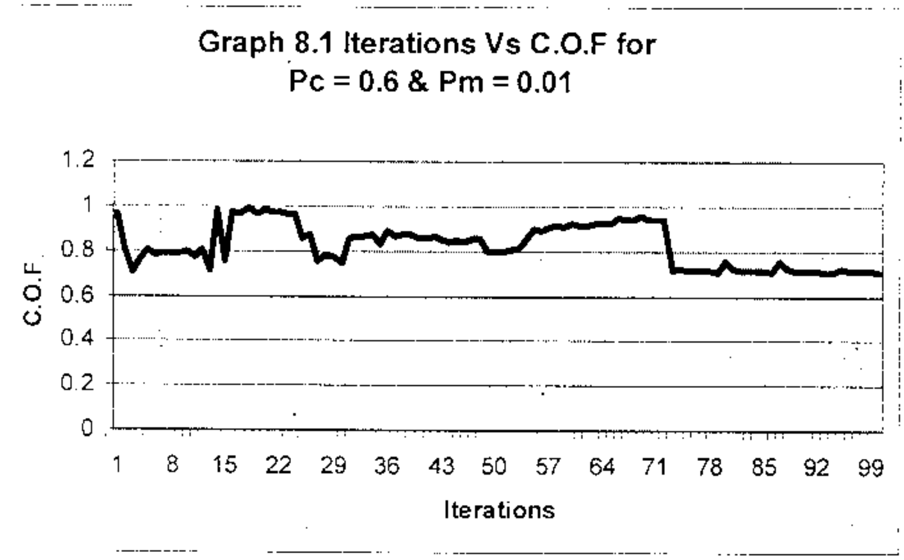


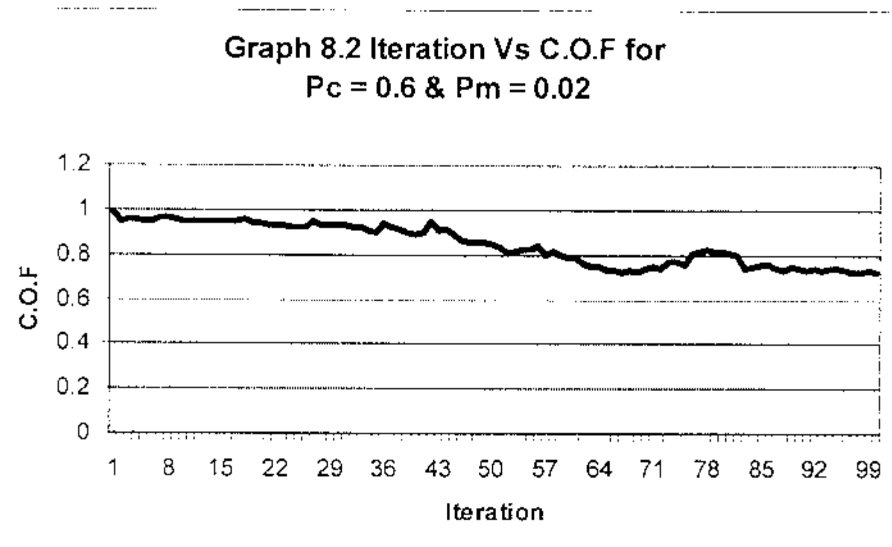
FIGURE 7.6 MUTATION OPERATIONS IN THE VB PROGRAM

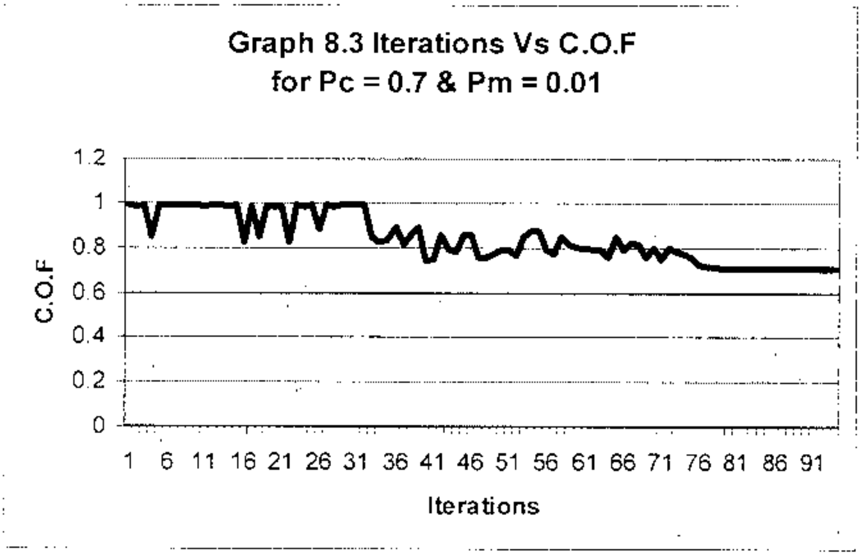
CHAPTER - 8

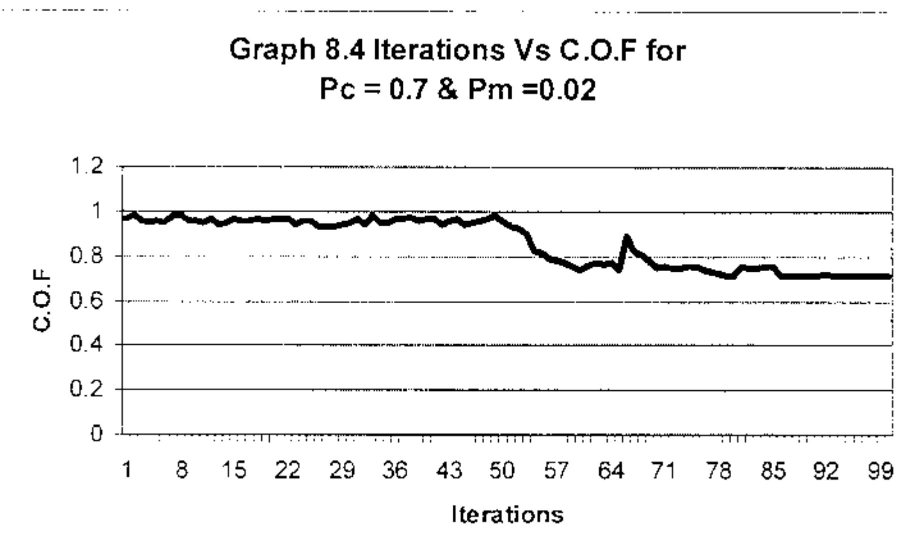
RESULT ANALYSIS AND VALIDATION

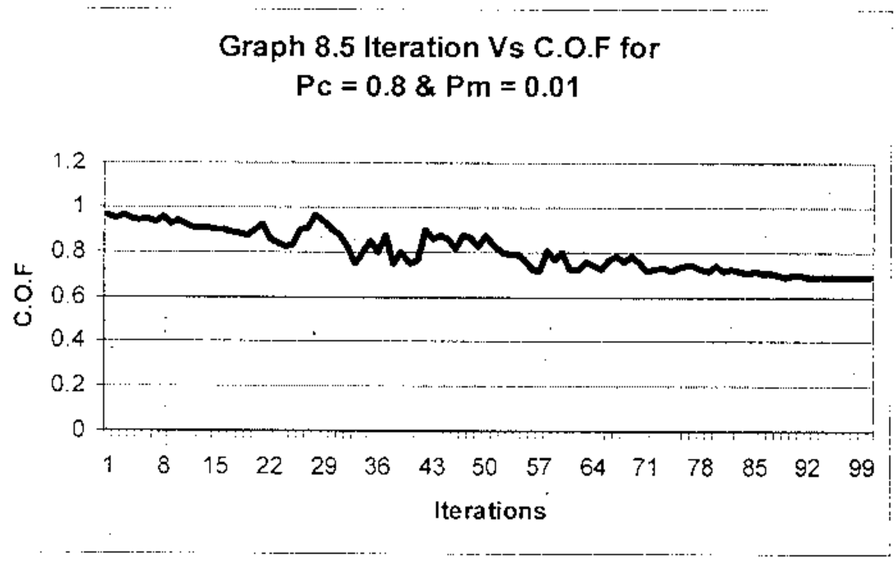
8.1 RESULT ANALYSIS

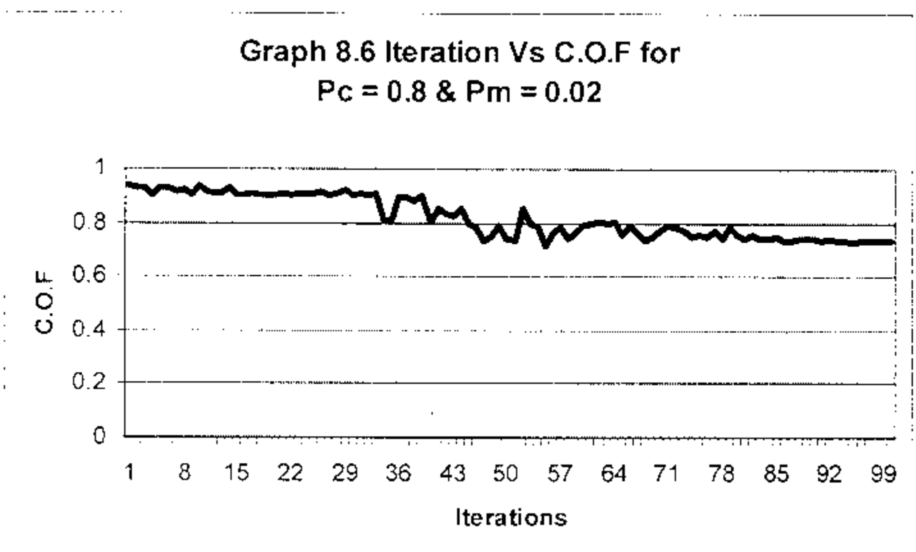


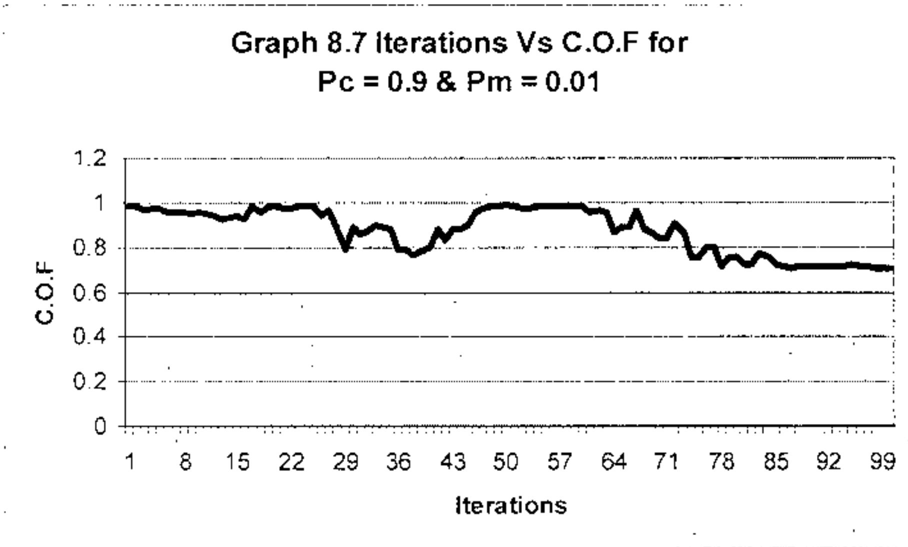


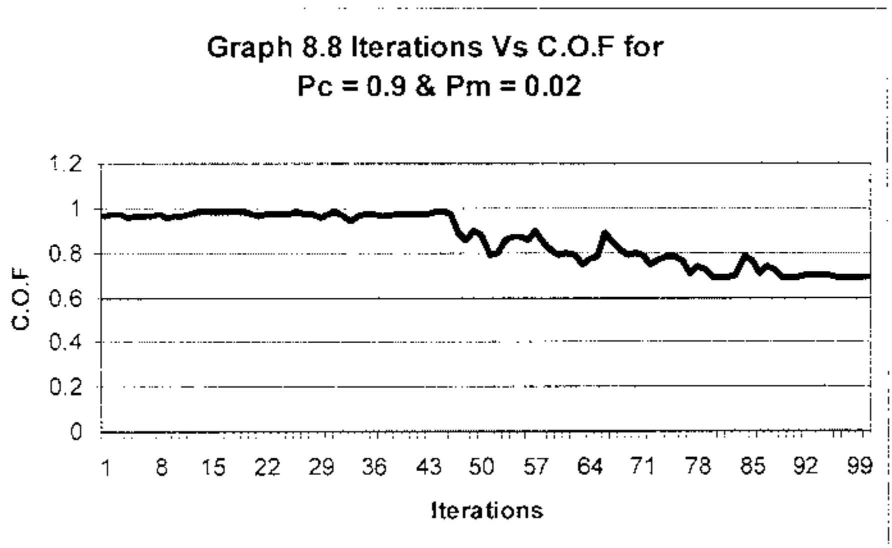

Graph 8.3 Iterations Vs C.O.F
for Pc = 0.7 & Pm = 0.01
1.2
1
0.8
0.6
0.4
0.2
0
C.O.F
1 6 11 16 21 26 31 36 41 46 51 56 61 66 71 76 81 86 91
Iterations


Graph 8.4 Iterations Vs C.O.F for
Pc = 0.7 & Pm =0.02
1.2
1
0.8
0.6
0.4
0.2
0
C.O.F
1 8 15 22 29 36 43 50 57 64 71 78 85 92 99
Iterations


Graph 8.5 Iteration Vs C.O.F for
Pc = 0.8 & Pm = 0.01
1.2
1
0.8
0.6
0.4
0.2
0
C.O.F
1 8 15 22 29 36 43 50 57 64 71 78 85 92 99
Iterations


Graph 8.6 Iteration Vs C.O.F for
Pc = 0.8 & Pm = 0.02
1
0.8
0.6
0.4
0.2
0
C.O.F
1 8 15 22 29 36 43 50 57 64 71 78 85 92 99
Iterations


Graph 8.7 Iterations Vs C.O.F for
Pc = 0.9 & Pm = 0.01
1.2
1
0.8
0.6
0.4
0.2
0
C.O.F
1 8 15 22 29 36 43 50 57 64 71 78 85 92 99
Iterations


Graph 8.8 Iterations Vs C.O.F for
Pc = 0.9 & Pm = 0.02
1.2
1
0.8
0.6
0.4
0.2
0
C.O.F
1 8 15 22 29 36 43 50 57 64 71 78 85 92 99
Iterations

Results obtained after the Genetic Algorithm concept to the proposed problem has been compared by varying the various operators such as cross over rate, mutation rate, number of generations, weightages for makespan, earliness and tardiness.

Percentage change in C.O.F. =

((Maximum value – Minimum value) /Maximum value)*100

From the graph of the minimum C.O.F. Vs Number of generations, various results obtained are tabulated below.

TABLE 8.1. ANALYSIS OF RESULTS

S.no	Cross over Rate	Mutation rate	Maximum C.O.F	Minimum C.O.F	Percentage C.O.F
1	0.6	0.01	0.9805	0.7034	28.26
		0.02	0.9919	0.7212	27.29
2	0.7	0.01	0.9940	0.7010	29.48
		0.02	0.9903	0.7156	27.74
3	0.8	0.01	0.9630	0.6864	28.72
		0.02	0.9353	0.7016	24.98
4	0.9	0.01	0.9848	0.7022	28.70
		0.02	0.9809	0.6903	29.62

From the tabulated results, it is clear that for cross over rate of 0.8, mutation rate of 0.01 and for 100 generations, minimum C.O.F value of 0.6864 is obtained having a 28.72% decrease in C.O.F value. So 0.6864 is the best C.O.F value and correspondingly the best schedule is,

09-01-07-04-06-03-08-10-02-05

Corresponding makespan value is = 4685 minutes

Corresponding earliness value is = 4980 minutes

Corresponding tardiness value is = 21340 minutes

The proposed methodology results in a C.O.F value of 0.6864 as compared to the maximum C.O.F value of 0.9948 for a crossover probability of 0.8 and mutation probability of 0.01.The percentage decrease of the C.O.F is 31%.

As per our analysis the preferred probability for crossover is 0.8 and probability for mutation is 0.01, which yield better results when compared with other values of probability. From the graph of C.O.F Vs number of iterations, after 100 iterations, the following results are observed,

- The C.O.F value keeps on varying and finally starts converging towards the steady state value.
- The graph indicates that the difference between the average C.O.F values and minimum C.O.F is greatly reducing till 100 iterations and beyond that. Therefore, it is clear that the nearest optimal solution has been reached within 100 iterations.

8.2. VALIDATION

The manufacturing industries are using traditional scheduling methods for scheduling the orders which are prone to be very less efficient when compared to newly emerged scheduling techniques using genetic algorithm, fuzzy logic, neural networks, tabu search, ant-colony algorithm etc. We took a study in the industry and found out that by using their traditional scheduling techniques for a certain number of jobs. their respective quantity, due date and time taken by the jobs in stage1 and in stage2, the schedule that was used by them was ,

09-06-02-08-05-01-10-03-04-07

At the end of the study we tabulated the completion time of each job in the sequence. For the due date given and the completion time of each job we calculated the values as follows,

Corresponding makespan value is = 8730 minutes

Corresponding earliness value is = 9810 minutes

Corresponding tardiness value is = 30035 minutes

For the same number of jobs, quantity, due date, time taken in stage1 and stage2, and for a crossover and mutation probability of 0.8 and 0.01 respectively, the best sequence and its values obtained by running the program giving the above details of the jobs as input is ,

09-01-07-04-06-03-08-10-02-05

By implementing this sequence for production of the jobs in the industry we arrived at the following results.

Corresponding makespan value is = 4685 minutes

Corresponding earliness value is = 4980 minutes

Corresponding tardiness value is = 21340 minutes.

This clearly shows that, the best sequence selected by running the VB program when used for scheduling jobs in the industry, the earliness, tardiness and makespan values are greatly reduced. This proves that by implementing this scheduling technique the industry can finish all its orders within the due date and also ensure that all its machines are used more efficiently.

CHAPTER 9

CONCLUSION AND SCOPE FOR FUTURE WORK

9.1. CONCLUSION

By using genetic algorithm an optimized solution has been obtained for the proposed problem. A software package has been developed for optimization of flexible flow shop scheduling by using the genetic algorithm approach. Also other enhancements have been made in the program which makes it possible to assign weightages for makespan, earliness, and tardiness as per requirement. Also any number of iterations can be retrieved for better results. But the solution extracted in this process is only approximate nearer solution to the required accuracy.

As the industry is scheduling the jobs randomly according to priority of jobs to be given to the customer, industry is facing problem of delay in delivery, and many uncertainties. So by following the proposed methodology an optimized solution is being achieved and industry could achieve the following objectives,

- Effective utilization of available resources.
- Total makespan is reduced.
- Total earliness is reduced.
- Total tardiness is reduced.
- Manufactured products could been delivered on due date.

9.2. SCOPE FOR FUTURE WORK

Manufacturing system considered in the above analysis can be expanded by adding more number of machines in different stages. Results obtained could be further optimized applying Fuzzy logic, Neural network and Hybrid genetic algorithm concepts. Also the following emerging new concepts Could be used to solve the similar kind of problem.

1. Simulated annealing
2. Tabu search
3. Ant colony algorithm

In future the following adversaries could be considered

- Any number of jobs can be scheduled for scheduling orders involving many jobs.
- Simulation can be done according to the shop floor capacities.
- Machine break downs can be considered and be incorporated in the program.
- Provision to incorporate rush and sudden orders in shop floor by assigning a priority value for the jobs to be delivered quickly.

REFERENCES

1. Bruno, J.L., E.G. Coffman, Jr. and R. Senthil (1974),Scheduling Independent Tasks to Reduce Mean Finishing Time, *ACM Communications 17*. pp 382- 387
2. Jagabandhu Sridhar (2002), A Genetic Algorithm Approach, *Scheduling in Flow Shop and Cellular Manufacturing Systems with Multiple Objectives*, pp. 17-32.
3. Lui Min, Wu dreng (1999), A Genetic Algorithm for Minimizing Makespan in case of Scheduling Identical Parallel Machines, *Artificial Intelligence in Engineering*, pp. 399-409.
4. Panneerselvam. R. (2001), *Production and Operations management*, Prentice Hall of India, New Delhi, pp. 256-269.
5. Rajasekaran. S, (2003), *Neural Networks, Fuzzy Logic and Genetic Algorithm Synthesis and Applications*, Prentice Hall of India, New Delhi, pp. 310-314.
6. Soh, C.K., (1996), *Fuzzy Controlled Genetic Algorithm Search for Shape Optimization*, pp. 170-198.