A
Major Project Report II
On

A Deadline Aware Approximation Algorithm for Scheduling Jobs With Burst Time and Priorities

Submitted in Partial Fulfillment of the Requirement

For the Award of the Degree of

Master of Technology

In

Computer Science and Engineering

By

Hitendra Pal
University Roll No. 2K13/CSE/06

Under the Esteemed Guidance of

Dr. S. K. Saxena
Computer Engineering Department, DTU

2013-2015

COMPUTER ENGINEERING DEPARTMENT

DELHI TECHNOLOGICAL UNIVERSITY

DELHI – 110042, INDIA

DECLARATION

We hereby declare that the major Project-II work entitled "**A Deadline Aware Approximation Algorithm For Scheduling Jobs With Burst Time And Priorities**" which is being submitted to Delhi Technological University, in partial fulfillment of requirements for the award of degree of Master Of Technology (Computer Science and Engineering) is a bonafide report carried out by me. The material contained in the report has not been submitted to any university or institution for the award of any degree.

Hitendra Pal

2K13/CSE/06

CERTIFICATE

This is to certify that the Major Project-II entitled **"A Deadline Aware Approximation Algorithm For Scheduling Jobs With Burst Time And Priorities "** is a bonafide record of work done by **Hitendra Pal, Roll No. 2K13/CSE/06** at **Delhi Technological University** for partial fulfilment of the requirements for the degree of Master of Technology in Computer Science & Engineering is a record of the candidate work carried out by him under my supervision.

Date: __ __ ___

(Dr. S.K.Saxena)
Project Guide
Department of Computer Engineering
Delhi Technological University

ACKNOWLEDGEMENT

First of all, I would like to express my deep sense of respect and gratitude to my project supervisor Dr. S.K.Saxena for providing the opportunity of carrying out this project and being the guiding force behind this work. I am deeply indebted to him for the support, advice and encouragement he provided without which the project could not have been a success.

Secondly, I am grateful to Dr. O.P.Verma, HOD, Computer Engineering Department, DTU for his immense support. I would also like to acknowledge Delhi Technological University library and staff for providing the right academic resources and environment for this work to be carried out.

Last but not the least I would like to express sincere gratitude to my parents and friends for constantly encouraging me during the completion of work.

Hitendra Pal

University Roll no: 2K13/CSE/06

M.Tech (Computer Science & Engineering)

Department of Computer Engineering

Delhi Technological University

Delhi – 110042

ABSTRACT

System's performance is highly affected by CPU Scheduling. Effective scheduling leads to better system performance. Process scheduling problem is a combinatorial problem in which arrangement of jobs is a key factor. Different algorithms and techniques have been developed, used to find the above factor, many techniques such as FCFS, SJF, Round-Robin, Priority, Multi-Level Queue and many more are applied but all these technique provide sequence of job relevant to their properties, the sequence which is necessary toward our requirement may be not full-filled by applying previously known techniques, to find such sequence its take exponential time.

In this project we proposed a method for process scheduling using a deadline aware approximation algorithm to provide efficient process scheduling, where required schedule has a certain weightage of priority and Burst time of job. In our problem we solve the desire sequence of jobs by using four approximation techniques.

To solve above problem in polynomial time there are approximation algorithms which solve the sequence of jobs in polynomial time. Genetic algorithm (GA) is one such approximation technique and another one is Artificial-Bee-Colony (ABC). Two further techniques are the modified-GA and modified-ABC.

In real life problems we have a deadline to complete certain set of jobs, we have to decide the sequence of job in which we complete our jobs before or nearby deadline, our proposed technique provide a way which leads toward a good solution, we known each job take some duration to execute (i.e. Burst time) and each job is associate with some priority, so considering both factor we developed a method in which we provide a certain weightage (requirement percentage) to priority, this lead to total completion time of jobs (Total-Turn-Around time) . Toward Total-Turn-Around time (TAT) we find a sequence of jobs.

We apply approximation algorithms to find desire schedule, our proposed CPU Scheduling method and comparison is made on the performance of CPU scheduling with and without the proposed technique in terms of number of iteration, number of test case, requirement percentage and tardiness (fitness value).

Keywords: CPU Scheduling, Optimization, Genetic Algorithm, Artificial-Bee-Colony and Crossover.

Table of Contents

List of Figures

List of Abbreviations

GA	Genetic Algorithm
MGA	Modified-Genetic Algorithm
ABC	Artificial Bee Colony
MABC	Modified-Artificial Bee Colony
CPU	Computer Processing Unit
HDD	Hard-Disk Drive
ID	Identification
FCFS	First-Come-First-Serve
SJF	Shortest-Job-First
NP-Complete	Non-Polynomial Complete
NP-Hard	Non-Polynomial Hard
SI	Swarm Intelligence
PSO	Particle Swarm Optimization
ACO	Ant Colony Optimization
OSI	Open System Interconnection
ATM	Asynchronous Transmission Mode
NO.	Number
REQ.	Requirement Percentage
I/O	Input-Output
TAT	Total Turn-Around Time

CHAPTER 1

INTRODUCTION

CPU scheduling is one of the most important functionality of the operating system, the scheduling of job play a vital role on the performance and efficiency of CPU. The major functionality of this task is to maximize the efficiency and finally enhance the performance. Scheduling of jobs should be done correctly and fairly, so that each job gets a chance to execute on the processor. Scheduling of jobs also plays a vital role as utilization of resources to a large extent depends on the efficiency of scheduling. The problem of getting a flexible or robust solution for scheduling problems is of uttermost importance for real-life applications.

The objective of multiprogramming is to reside multi-process in main memory, in today time multi-programming is similar to multi-tasking, which leads to maximize CPU utilization. The objective of time sharing is all the processes are run concurrently and the CPU is switched among processes so frequently. In a uni - processor system only one process is running at a time. In our problem we consider one process is executing at a time. The work of a dispatcher is allocating a CPU to a process which is decided by a scheduler to be processed under allocation. Scheduling is a factor which majorly affects the performance of the system, because it decides which process will come under execution and which will wait.

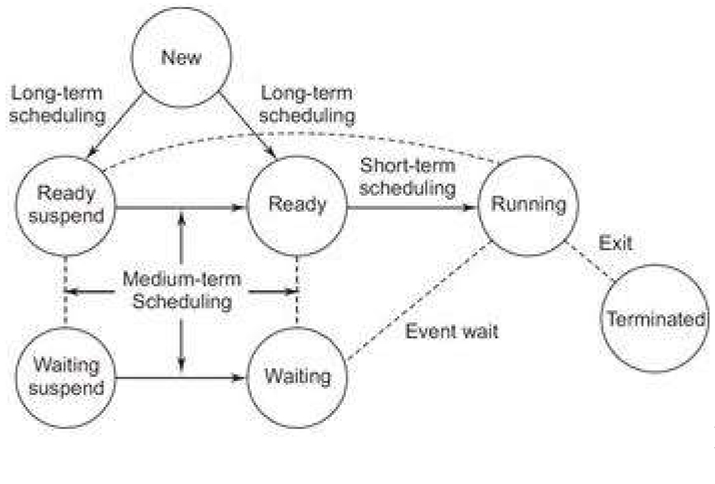

Figure.1.1 [31]

In above figure.1 all the states of process are shown, the above representation and linking of path shows how a program is to move from one state to another with or without external effects.

New State: when we double click on program, program is added to the job-pool which is in HDD and unique ID is associated with that process.

Ready State: process is being to execute.

Execute State: process is under execution, the CPU is allocated to it.

Waiting State: process is waiting for input/output, completion, this state is in main memory.

Suspend State: Process is temporary swap-out and place in that part of HDD called swap space.

Terminate State: Process has fairly finished its execution

Types of Scheduling

- **Long-term Scheduling**
  When a new process is created and is added to the job-pool which is in HDD and unique ID is associated with that process. The long-term scheduler decides which programs are now admitted into the system for processing and which are later admitted into the system. The work of admitting jobs is done by the dispatcher. The long-term scheduler is lesser frequent then Medium -term scheduler.

- **Medium-term Scheduling**
  Medium-term scheduling is scheduled which decides which process is swap-in and which is swap-out. Virtual memory is combination of main memory and part of HDD called swap space. The process of the swap-in, swap-out is handled by Medium-term scheduler. The medium-term scheduler is lesser frequent than a Short-term scheduler.

- **Short-term Scheduling**
  Short-term scheduling is scheduled which decides which process is coming under execution from ready queue, short-term scheduler decides which process is now bring

under execution and this work is done by dispatcher. It means decisions are made by scheduler and dispatcher fulfills it. The short-term scheduler executes most frequent scheduler.

Scheduling Criteria

- **CPU Utilization:**
  The amount of time CPU is busy in performing useful task divide by total amount of time. At 99 % of jobs, 99 % of the time is spent on a particular bunch of lines. To increase utilization the page which consist those lines much always be in main memory.

- **Throughput:**
  Throughput is the rate at which number of processes are completed their execution per unit time.

- **Turnaround time:**
  It is a total time when a process is arrive on ready state till the process is complete, its execution and move to terminate state.

- **Waiting time:**
  Waiting time is the total time spent by process in ready queue.

- **Response time:**
  Response time is the time when a process is brought into main memory and a time it spent first time in ready queue, that time is contribute in response time.

- **Fairness:**
  CPU should be allocated to each process in a fair share manner.

- **Tardiness:**
  It is time which is difference of job completion time and its deadline time.

- **Makespan:**

When several jobs are together competing of execution, then the starting of first job time to the end of last job run that time is contributing to makespan.

There are two modes of scheduling

Preemptive Scheduling:
In preemptive, the scheduling criterion is based on remaining time on the process currently being processed. The process with shortest remaining burst time is executed / scheduled first then other processes. In other words, if a new process arrives, then the remaining burst time of the current process is compared with the burst time of new process, whichever is shortest, is scheduled and the other process is made to wait.

Non-preemptive Scheduling:
In non-preemptive, if an arriving process has a shorter burst time, then the executing process, the process in execution is not preempted, but is allowed to finish. In our paper, we use preemptive (Shortest remaining time first) scheduling algorithm.

Process scheduling problem is a combinatorial problem in which arrangement of jobs is a key factor. Scheduling of jobs is a combinatorial problem which consumes exponential time. To solve the above problem in polynomial time there are approximation algorithms which solve the sequence of jobs in polynomial time. Genetic algorithm (GA) is one such approximation technique and another one is Artificial-Bee-Colony (ABC).

Genetic algorithm is an approximation base algorithm which is inspired by nature. It was first developed by Holland [2] and De Jong [3]. The genetic algorithm is applied to varying field of problems which are non-polynomial (NP-Complete or even NP-Hard). GA play a vital role in solving them in polynomial time.

Scheduling of jobs is a combinatorial problem which consumes exponential time but GA applied on many scheduling problems to solve them in polynomial time.

A basic GA consists of five components. These components are as follows: first, a random number generator ; second, a fitness evaluation unit ; last three are the genetic operators Selection(Reproduction), Crossover and Mutation operations.

1.1. Scheduling Algorithms

Scheduling algorithms or scheduling policies are used for short-term scheduling. The main functionality of short-term scheduling is to allocate a processor in such a way as to optimize one or more aspects of scheduling criteria / system behavior.
Let say we have a uniprocessor system for these scheduling algorithms. Scheduling algorithms decide which of the process in the ready queue is to be allocated to the processor is the basis of the type of scheduling policy and whether that scheduling policy is either preemptive or non-preemptive in nature. For scheduling several aspects such as arrival time, burst time and priority are also playing a role in scheduling of jobs.
List of scheduling algorithms is as follows:

- **First-come-first-served scheduling (FCFS) algorithm.**
- **Shortest Job First Scheduling (SJF) algorithm.**
- **Shortest Remaining time First (SRTF) algorithm.**
- **Non-preemptive priority algorithm of scheduling.**
- **Preemptive priority algorithm of scheduling.**
- **Round-Robin algorithm of scheduling.**

For explaining some of the above scheduling let us take an example.

	Arrival time	Burst time	Priority
Process1	0	4	2
Process2	3	6	1
Process3	5	3	3
Process4	8	2	1

Table.1.1

1.1.1. First-come First-served Scheduling (FCFS)

As the name refers First-come First-served Scheduling follow first enter, first leave out method. Each process is waiting in ready queue for the allocation of CPU. When the currently running process finished its execution, the next selected process for execution is the oldest process in the Ready queue. That is the first entered process among the available processes in the ready queue. The average waiting time for FCFS is quite long. It is non-preemptive in nature.

TURN-AROUND TIME=WAITING TIME + BURST TIME

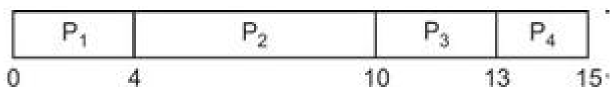

Table 1.2

Advantages

- Better for long processes, if we consider context switching overhead.
- Simple method (i.e., minimum overhead on processor)
- Minimum context switching.
- No starvation

Disadvantages

- Waiting time of shorter process is more if it arrives later higher burst time process.
- Throughput is poor, the higher burst time process comes early then lower burst time process.

1.1.2. Shortest Job First Scheduling (SJF)

Shortest-Job-First scheduling algorithm is of two types: i) preemptive (Shortest remaining time first) and ii) non-preemptive [12] in nature. In preemptive, the scheduling criterion is based on remaining time on the process currently being processed. The process with shortest remaining burst time is executed / scheduled first then other processes. In other words, if a new process arrives, then the remaining burst time of the current process is compared with the burst time of new process, whichever is shortest, is scheduled and the other process is made to wait.

In non-preemptive, if an arriving process has a shorter burst time, then the executing process, the process in execution is not preempted, but is allowed to finish. In our paper, we use preemptive (Shortest remaining time first) scheduling algorithm.

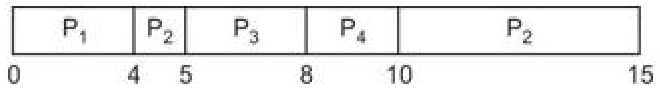

Table 1.3

Advantages

- Total-turn-around time is better.
- Throughput is good.

Disadvantages

- Typical to implement, difficult to get exact burst time priori.
- Starvation is possible for the longer processes, if shorter is continuously arriving.

1.1.3. Priority Scheduling

Every CPU scheduling algorithm can be considered as a priority algorithm because the property which follow one's algorithm can be treated as priority. The priority scheduling algorithm is preemptive [12] in nature in which processes are scheduled according to their priority, whereas highest priority job can run first whereas lower priority job can be made to wait. The priority of a given particular process is subjective and it is determined by many factors, in our algorithm 0(zero) denotes the highest priority and 5 denotes the lowest priority. The biggest problem of this algorithm is the starvation of a process [11].

Non-preemptive Priority Scheduling

In this type of scheduling the CPU is allocated to the next highest priority process after completing the present running process.

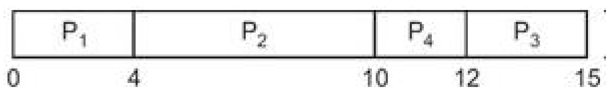

Table 1.4

Advantage

- Less response time for the highest priority processes.

Disadvantage

- Starvation can be possible for the lowest priority processes.

Preemptive Priority Scheduling

In this scheduling if new arriving process has higher priority than a process which is under execution, executing process is preempted and newly process is coming under allocation of CPU.

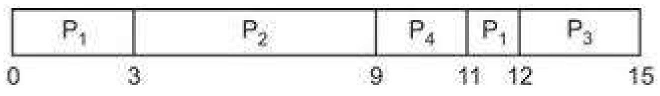

Table 1.5

Advantage

- Very good response time for the highest priority process over non-preemptive version priority CPU scheduling.

Disadvantage

- Starvation can be possible for the lowest priority processes.

1.1.4. Round-Robin Scheduling

This type of scheduling algorithm is basically designed for multi-processing / time sharing system. It is similar to the FCFS scheduling algorithm if we add preemption on it, preemption is based on time-slice. Round-Robin Scheduling is also called as time-slicing (time quantum)scheduling. Clock interrupt is generated at periodic interval of 5-50ms. When the

interrupt occurs, the next ready job is selected on a FCFS (First-come-First-serve) basis and the current running process is moved in the ready queue. This process is known as time-slicing (time quantum) scheduling, because each process is executed equal to time quantum before it being preempted.
One of the below happens:

- The burst time of the process may be smaller than the time quantum.
- Burst time of currently executing process is longer than the time quantum. In this case the context switch occurs and the currently running process is put at the tail of the ready queue.

Mostly, the duration of time-slice is decided on the basis of 90% of jobs can finish in their first time-slice period, this is because to reduce the number of context switching's.

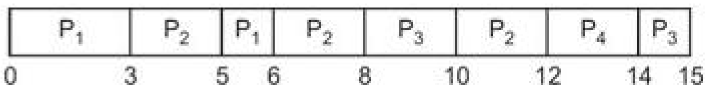

Table 1.6

Advantages

- Best response time.
- Highly interactive scheduling algorithm.

Disadvantages

- Choosing quantum value is crucial.
- Number of context switch is more.
- If time quantum is very small, throughput is low.

1.2. Approximation Algorithm

1.2.1 Genetic Algorithm

Genetic algorithm is an approximation base algorithm which is inspired by nature. It was first developed by Holland [2] and De Jong [3]. The genetic algorithm is applied to varying field of problems which are non-polynomial (NP-Complete or even NP-Hard). GA play a vital role in solving them in polynomial time.

Scheduling of jobs is a combinatorial problem which consumes exponential time but GA applied to many scheduling problems to solve them in polynomial time.

A basic GA consists of five components. These components are as follows: first, a random number generator ; second, a fitness evaluation unit ; last three are the genetic operators Selection(Reproduction), Crossover and Mutation operations.

The basic algorithm is summarized below:

```
1: Initialize Population
2: repeat
3:    Evaluation
4:    Selection
5:    Crossover
6:    Mutation
7: until requirements are met
```

Initialize Population

It is a starting phase of Genetic algorithm in which initial population is generated, each element of the population is called Chromosome. A chromosome consists of genes, to represent a gene we required some encoding schemes. Various types of encoding schemes are:

- Binary encoding
- Permutation encoding
- Value/Real encoding
- Octal & Hexadecimal encoding

In our algorithm we use the Value encoding scheme.

Evaluation

It is phased in, which evaluation of the problem is done as a result. There are is a function which is treated as a result evaluation, i.e. objective function / fitness function. Fitness function can be defined by many factors such as tardiness, makespan, total-turn-around time, total-waiting time.

In our problem fitness function is tardiness.

Selection

It is phase of the selection of two parent chromosome from the set of initial population. On these two chromosomes the further lower phases (Crossover, Mutation) of Genetic algorithm are applied.

Crossover

It is the phase which is applied on selected two parent chromosome, There are several types of crossover:

- One-point crossover.
- Two-point crossover.
- Half uniform crossover and Uniform crossover.
- Ring crossover.
- Hybrid One-Two point crossover

One-point crossover

Select two individuals randomly from the current population for crossover operation. Apply one-point crossover operation on the parent chromosome to generate new offspring chromosome. A new set of sequence vector is generated for new offspring. Compute the cost for that offspring; Compute the fitness of updates individual. Replace the worst parent and associated chromosome with new best offspring and its chromosome if it is better update individuals.

10100011	00110**110**	**10100110**	00110011
Parent1	**Parent2**	**Offspring1**	**Offspring2**

Table 1.7

Two-point crossover

Select two individuals randomly from the current population for crossover operation. Apply two-point crossover operation on the parent chromosome to generate new offspring chromosome. A new set of sequence vector is generated for new offspring. Compute the cost for that offspring; Compute the fitness of updates individual. Replace the worst parent and associated chromosome with new best offspring and its chromosome if it is better update individuals.

101000**11**	00**1101**10	**10110111**	00100010
Parent1	**Parent2**	**Offspring1**	**Offspring2**

Table 1.8

Hybrid One-Two point crossover

Select two individuals randomly from the current population for crossover operation. Apply two-point crossover operation on the parent chromosome to generate new offspring chromosome. Compute the fitness of that offspring; now apply one-point crossover operation on the parent chromosome to generate new offspring chromosome. Compute the fitness of that offspring. Compare both pairs of fitness of one-point and two-point; select one pair which is minimum of both. Replace the worst parent and associated chromosome with new best offspring and its chromosome if it is better update individuals.

If offspring after one-point crossover have better fitness than offspring after a two-point crossover.

10100011	00110**110**	**10100110**	00110011
Parent1	**Parent2**	**Offspring1**	**Offspring2**

Table 1.8

If offspring after two-point crossover have better fitness than offspring after a one-point crossover.

101000**11**	00**1101**10	**10110111**	00100010
Parent1	**Parent2**	**Offspring1**	**Offspring2**

Table 1.9

Uniform crossover

Select two individuals randomly from the current population for crossover operation. Take an auxiliary vector of equal number of genes in chromosomes and filled by randomly 0 and 1. All genes of both chromosomes having indexed same to the index of the auxiliary vector with 0 values are swapped. Compute the fitness of updates individual. Replace the worst parent and associated chromosome with new best offspring and its chromosome if it is better update individuals.

Ring crossover

Select two individuals randomly from the current population for crossover operation. Apply ring crossover operation on the parent chromosome to generate new offspring chromosome. In ring crossover assume both parent chromosomes are in two arrays now, imagine left end of one array is joining with the left end of another array and right end of one array is joining with right end of another array generate random number **r** b/w 0 & number of gene in chromosome, perform circular rotation r times. Arrays after the perform above step are new offspring chromosome. Compute the fitness of updates individual. Replace the worst parent and associated chromosome with new best offspring and its chromosome if it is better update individuals.

Suppose **r** is 3.

10100011	00110110	**10010100**	10110110
Parent1	**Parent2**	**Offspring1**	**Offspring2**

Table 1.10

Mutation

If mutation criteria is met(it is randomly 3 times out of 100 times) then ,Select two individuals based on minimum fitness value from the current population of mutation

operation. Apply mutate operation to generate new chromosomes (new offspring).Compute the fitness value for that offspring, Replace the worst parent and associated chromosome with new best offspring and its chromosome if it is better update individuals.
It is the phase which is applied on selected two parent chromosomes, There are several types of mutation:

- Move
- Swap
- Move and Swap
- Rebalancing

Move Mutation

Select two individuals based on minimum fitness value from the current population of mutation operation. Apply mutate operation (move) to generate new chromosomes (new offspring).Compute the fitness value for that offspring, Replace the worst parent and associated chromosome with new best offspring and its chromosome if it is better update individuals.

0011011**0**	0011011**1**
Parent	**Offspring**

Table 1.11

Swap Mutation

Select two individuals based on minimum fitness value from the current population of mutation operation. Apply mutate operation (move) to generate new chromosomes (new offspring).Compute the fitness value for that offspring, Replace the worst parent and associated chromosome with new best offspring and its chromosome if it is better update individuals.

10100**0**11	00110**1**10	10100**1**11	00110**0**10
Parent1	**Parent2**	**Offspring1**	**Offspring2**

Table 1.12

1.2.2. Artificial Bee Colony Algorithm

In ABC algorithm, we imitate all these tasks in a computerized way. Employee starts the process of collecting nectars from different food sources. They bring back the nectars to the hive and exchange information about the food source they visited by a particular dance called waggle dance. Physical interpretation of waggle dance is, more a bee dances higher the quality of the food source visited by it. After seeing the waggle dance, onlooker bee decides the food source they are going to visit and this process continues until a food source is dead. When a food source is dead, it is abandoned, and scout bee starts searching for a new food source to replace the abandoned one. The benefit of such social organizations is the increased flexibility to adapt to the changing environments. The bee members use a sophisticated communication protocol with bee-to-bee signals, a stigmergic feedback cue for bee-to-group or group-to-bee interaction.

In ABC algorithm, colony bees are divided into three groups: employed bees, onlooker bees and scout bees. The number of employees is equal to the number of food sources i.e. number of solutions in the population. The employee whose food source is exhausted is provided with a new food source by the scout bee. Each food source position is equivalent to a potential solution of optimization problem, with its quality or fitness value acting as the nectar amount. Whenever any bee finds a food source, it signals the other bees by stigmergy (waggle dance), the quality and the location of the food source. This attracts a large number of bees (onlooker bees) towards good food sources for further exploration.

The main steps of the algorithm are as below:

```
1: Initialize Population
2: repeat
3: Place the employed bees on their food sources
4: Place the onlooker bees on the food sources depending on their nectar amounts
5: Send the scouts to the search area for discovering new food sources
6: Memorize the best food source found so far
7: until requirements are met
```

The ABC algorithm has three operational phases:

- Scout bees do random search for the food, & find near optimal food sources, completely random.
- Employed bees visit food source and gather information about food source location and the quality. They have memory of the places they have visited before and quality of food there, & performs the local search to try to exploit the neighboring sources to locate the best.
- Onlooker bees wait in the dance area to decide which food source is better on the basis of information provided by employees. They perform the global search for discovering the global optimum.

1.3. Motivation

CPU scheduling is one of the most important functionality of the operating system, the scheduling of job play a vital role on the performance and efficiency of CPU. Scheduling of jobs plays a vital role as utilization of resources to a large extent depends on the efficiency of scheduling. The problem of getting a flexible or robust solution for scheduling problems is of uttermost importance for real-life applications. The major functionality of this task is to maximize the efficiency and finally enhance the performance.

Process scheduling problem is a combinatorial problem in which arrangement of jobs is a key factor. Different algorithms and techniques have been developed, used to find the above factor, many techniques such as FCFS, SJF, Round-Robin, Priority, Multi-Level Queue and many more are applied but all these technique provide sequence of job relevant to their properties, the sequence which is necessary toward our requirement may be not full-filled by applying previously known techniques, to find such sequence its take exponential time.

Lot of research has been done in the area of optimization of CPU scheduling, so GA plays a vital role in finding nearby optimal solutionsof optimization and search problems in polynomial time[4]. The guided randomsearch based algorithms typically generate multiple candidate solutions, sampled from a feasible solution space and uses a guided search for exploration. A host of evolutionary, swarm intelligence techniques like*Genetic algorithms*

[1], [13], *Particle Swarm Optimization*[14], *Ant Colony Optimization* [16],*Simulated Annealing* [15], and *Tabu search* [17], [18]have been successfully applied to the scheduling.

- In real life problems we have a deadline to complete certain set of jobs, we have to decide the sequence of job in which we complete our jobs before or nearby deadline, our proposed technique provide a way which leads toward a good solution, we known each job take some duration to execute(i.e. Burst time) and each job is associate with some priority, there are many other factor associated with job but these two are important. So considering both factor we provide a sequence of jobs which is towards desired optimal

1.4. Research Objective

With the motivation explained in the previous section, the objective of our research work can be identified as:

- Development of a mechanism when there are certain number of jobs (any real life work is treated as job) with number of parameter associated with it, provide a sequence of jobs which is towards desired optimal.
- In this research we consider two factor of job, i.e. burst time and priority of job.
- To improvise the completion time of set of jobs we provide set of requirement percentage of priority, user can choose particular requirement percentage of priority as per his/her deadline.
- Let there be N Preemptive processes (jobs) (j_1,j_2,...,j_n) [12] that are waiting to be processed by a single processing system. Each job j_i has its arrival time a_i, burst time b_i and priority p_i respectively.
- We have two schedules of N processes, one ordered by their burst times and the other according to their priorities and a third schedule needs to be generated that is a weighted sum of both priority and Shortest–Job-First(SJF).

1.5. Major Project Organization

We start this dissertation with introduction in chapter 1. A detailed description of background is presented in chapter 2 which includes scheduling & its applications, literature review of Optimization Algorithm. Chapter 3 explains about proposed problem statement and its proposed solution. Chapter 3 also gives a brief about the optimization technique we have used. Chapter 3 also explains in detail about our proposed algorithm Modified-GA and proposed Hybrid-ABC. We evaluate the performance of the proposed algorithm and technique with CPU scheduling in chapter 4. We conclude about the work done and observations in chapter 5.

CHAPTER 2

LITERATURE REVIEW

2.1. Scheduling and its Application

Scheduling is key factor to perform any work in a systematic way. In real life everyone is surrounded with ample of work and have to perform each work completely either in efficient way or inefficient way, but our objective is to perform that work in efficient and for that we highly recommend for scheduling.

Scheduling of job play a vital role on the performance and efficiency of CPU. The major functionality of this task is to maximize the efficiency and finally enhance the performance. Scheduling of jobs should be done correctly and fairly, so that each job gets a chance to execute on the processor. Scheduling of jobs also plays a vital role as utilization of resources to a large extent depends on the efficiency of scheduling. The problem of getting a flexible or robust solution for scheduling problems is of uttermost importance for real-life applications.

If we say, we have to run our life without scheduling then there is no deadline of any work, then there is no systematic in nature and without it life cannot be run. So, scheduling play a key factor in the real world. The application of scheduling is everywhere in each and every field.

Some of the application, we mention below:

2.1.1. Transport system

- **Railway System**
  If there is no scheduling of arrival and departure of train then there shell be a deadlock or even a crash.
- **Airline System**
  same as railway system, If there is no scheduling of arrival and departure of train then there shell be a deadlock or even a crash.

2.1.2 Education System

- **Library Management System**
  If books of every department is not separately schedule and stored randomly then time to search for particular book is unpredictable.
- **Examination Management System**
  in this section also scheduling is play a vital role, if scheduling is not follow, then examination never be able to conduct or if conducted, it is in such manner which doesnot make any importance on examination result.

2.1.3. Banking Sector

- Opening / closing time of banks.
- Scheduling of transaction.
  - Inconsistency in data.
  - Non-Atomicity.
  - Non-durability.

2.1.4. Computer Science & Engineering

- **Graph Coloring Problem**
  The beauty of this problem is schedule nodes is such a way no two adjacent node are come under same set.
  In graph coloring problem we have graph with its nodes and edges, we have to color each node with minimum number of colors and constrain of coloring are as follows:
  - Make minimum numbers of independent set.
  - In independent set, all vertices are non-adjacent to each other.
  - The number of independent set is equal to minimum number of colors required to perfectly color a graph.

- **A service architecture for ATM.**

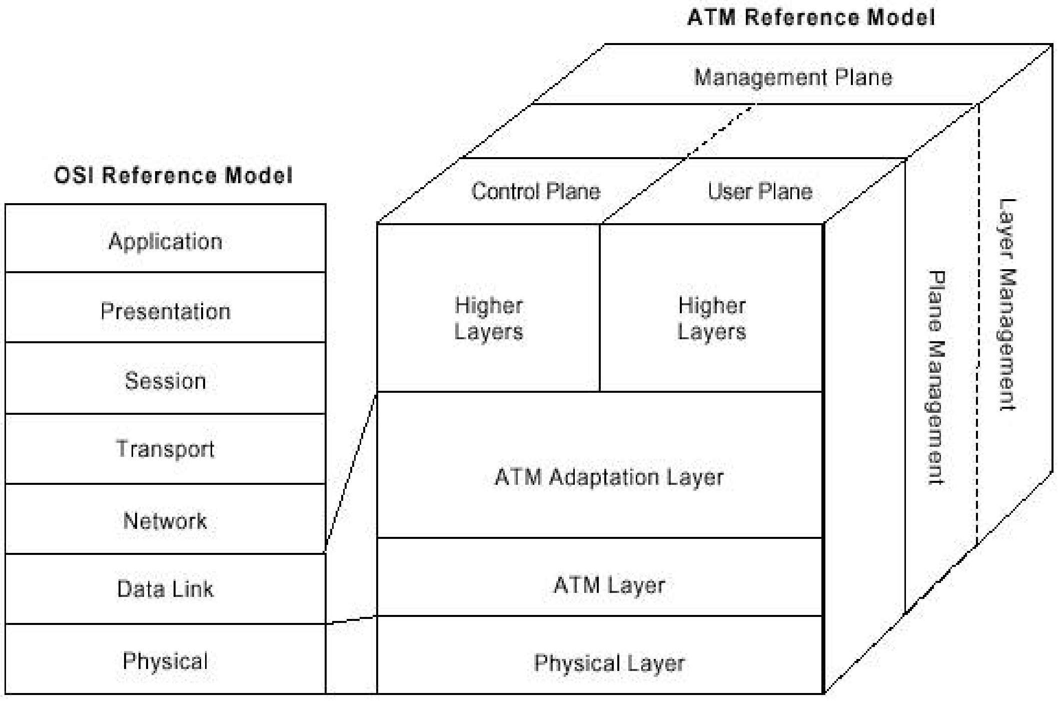

Figure: 2.1 [19]

2.1.5. CPU Scheduling

Process scheduling problem is a combinatorial problem in which arrangement of jobs is a key factor. Different algorithms and techniques have been developed, used to find the above factor, many techniques such as FCFS, SJF, Round-Robin, Priority, Multi-Level Queue and many more are applied but all these technique provide sequence of job relevant to their properties, the sequence which is necessary toward our requirement may be not full-filled by applying previously known techniques, to find such sequence its take exponential time. System’s performance is highly affected by CPU Scheduling. Effective scheduling leads to better system performance.

The problem of getting a flexible or robust solution for scheduling problems is of uttermost importance for real-life applications. Scheduling of jobs also plays a vital role as utilization of resources to a large extent depends on the efficiency of scheduling. The major functionality of this task is to maximize the efficiency and finally enhance the performance. Scheduling of jobs should be done correctly and fairly, so that each job gets a chance to execute on the processor. CPU scheduling is one of the most important functionality of the operating system, the scheduling of job play a vital role on the performance and efficiency of CPU.

Scheduling of jobs also plays a vital role as utilization of resources to a large extent depends on the efficiency of scheduling. The objective of multiprogramming is to reside multi-process in main memory, in today time multi-programming is similar to multi-tasking, which leads to maximize CPU utilization. The objective of time sharing is all the processes are run concurrently and the CPU is switched among processes so frequently. In a uni - processor system only one process is running at a time.

In preemptive, the scheduling criterion is based on remaining time on the process currently being processed. The process with shortest remaining burst time is executed / scheduled first then other processes. In other words, if a new process arrives, then the remaining burst time of the current process is compared with the burst time of new process, whichever is shortest, is scheduled and the other process is made to wait.

In non-preemptive, if an arriving process has a shorter burst time, then the executing process, the process in execution is not preempted, but is allowed to finish. In our paper, we use preemptive (Shortest remaining time first) scheduling algorithm.

2.2 Genetic Literature

This section presents a comprehensive study of Genetic Algorithms (GA) applied to scheduling in last 2 to 3 decades. GA is widely used in many engineering fields and it is an effective, it's optimization performance was verified in [1]. GA's was developed by Holland [2] and De Jong [3] , based on mechanics of natural selection in the biological system.

GA has been applied to different application areas, specifically in combinational problems, such as scheduling [4-9]. In general ,scheduling problems have been proved to be NP-Complete [7] and so to find an optimal solution it take exponential time, which is not feasible in real time applications with time constraints, so GA plays a vital role in finding nearby optimal solutions of optimization and search problems in polynomial time[4].

2.3 Artificial Bee Colony Literature

In swarm based optimization technique, the bees based optimization technique are population based search algorithm that mimics the food foraging behavior of honey bees. This technique was first proposed in 2005 by D.T.Pham [22] and Karaboga [23] independently. D.T.Pham [22] was proposed ABC on the basis of foraging behaviour of honey bee, Karaboga [23] was also proposed ABC on the basis of foraging behaviour of haney bee but it was more effective in searching for food source.

- [24] M. S. Kiran and M.G. Unduz, In this paper author present a new approach of artificial bee colony algorithm for solving the numerical optimization problems in which exploitation and local search abilities were improved.
- [27] DENG Guanlong, XU Zhenhao and GU Xingsheng, In this paper, author present a blocking flow shop scheduling problem in which the role is to minimize the total flow time, by designing some new schemes for all three phases (employee bee, onlooker bee and scout bee phase).
- [28] A. Madureira, I. Pereira and A. Abraham, In this paper, author present a Job-shop scheduling optimization technique in which the role is to minimize the total tardiness of jobs with constrain we have only one machine on which all operation have to run.
- [29] M. S. Kiran and A. Babalik, In this paper author proposed a improved version of artificial bee colony algorithm for continues optimization problem in which selection of neighborhood of the candidate solutions is done in onlooker bee phase on the bases of information shared by the employed bees.
- [30] Ana Madureira, Bruno Cunha and Ivo Pereira, In this paper author present a way of intercommunication for distributed scheduling, it is based on self-organized scheduling system. In this agents are collaborate together to improve their local solution and the global schedule. The cooperative mechanism is generated by resource agents to analyze the schedule and idle time is reduce by the self-organized behaviour of artificial bee colony algorithm.

2.4. Some Other Approximation Techniques

2.4.1. Ant Colony Optimization

ACO is a optimizing probabilistic technique which is used to find a good path toward solution of computational problems using graph. We known that there are many computational problems which are graph based and if we solve them in greedy way it take enormous exponential time. ACO is initially proposed by Marco Dorigo [21] in 1992 in his PhD thesis.

A basic algorithm is summarized:

1. Initialize pheromone.
2. Single ant deposited pheromone perceivable by other ants.
3. While moving from nest to food-source (or vice-versa) pheromone is lay.
4. Guidance for other foragers to food source.
5. Even shortest way to food source is optimized.

Foraging behavior of ants

- **Pheromone**
  Pheromone is a substance which is release by an ant to give an direction to other ants, the path which have higher percentage of pheromone that path is followed by ants and during following path ants release pheromones, so the concentration of pheromone is slightly increased in single path and that become the shortest path.

Recent work of ACO algorithm on scheduling

- [25] R. Chaukwale and S. S. Kamath, in this paper ACO play its role for local balancing in job-shop scheduling problem. Job-shop scheduling problem is NP-Hard problem, where number of jobs are N (j_1,$j_{2,}$…,j_n), number of machines are M ($m_{1,}$$m_2$,…,$m_n$) and each process have M operation. We have to find a schedule each operation of all the jobs must be run on any of the machine in such a way, total makespan (completion time) should be less. So, here ACO make a balance to operation in such a way the total makespan (completion time) is nearby optimal.
- [26] Tawfeek, M.A., El-Sisi, in this paper, ACO play its role for optimizing cloud task scheduling. Cloud collection of software as a service, data as a service, in which we do-not need to install software on our local drive and we use the services of software. Cloud computing is a collection of interconnected and virtual computer system, scheduling a task in cloud is also a NP-Hard problem, here ACO provide a solution which is non-exponential and nearby optimal.

2.4.2. Particle Swarm Optimization

In PSO is an optimizing probabilistic technique which is used to find a best food source out of number of food source. Particle swarm optimization was invented by Kennedy and Eberhart [20] Conversing rate of PSO is faster than ACO but do not guarantee optimal optimization all the time. We know that there are many computational problems which are graph based and if we solve them in greedy way it take enormous exponential time. In PSO all population particle are moving in search space. Particles are moving toward the direction of current optimum particles and changing their position to move towards the optima. In every iteration there is a current particle achieve called best particle P_{best} , each particle follow it and best of the population is the global best G_{best}.

In PSO there are two important keys:

- **Position of particle**
  The new position of the particle is calculated by previous position of particle and the current velocity of particle. In this way particles are move towards best particle.
- **Velocity of particle**
  New velocity is calculated by the previous velocity with some parameter called inertia weight and the difference of old position to new position of particle with some constant factor which determine the significance of P_{best} and G_{best}.

The basic algorithm is summarized below:

```
1: Initialize Population
2: repeat
3: Calculate fitness values of particles
4: Modify the best particles in the swarm
5: Choose the best particle
6: Calculate the velocities of particles
7: Update the particle positions
```

CHAPTER 3

PROPOSED WORK

3.1. Problem Statement

Let there be N Preemptive processes (jobs) ($j_1,j_2,...,j_n$) [12] that are waiting to be processed by a single processing system. Each job j_i has its arrival time a_i, burst time b_i and priority p_i respectively.
The objective is to find the schedule that satisfies the following constraints.

1. CPU should process every job.
2. Processes are independent and compete for resources.
3. We have two schedules of N processes, one ordered by their burst times and the other according to their priorities and a third schedule needs to be generated that is a weighted sum of both priority and Shortest–Job-First(SJF).

3.2. Proposed Solution

We have two schedules one according to SJF and the other according to priority, we know that SJF gives us optimal Total Turnaround Time(TAT) but if we consider only it we neglect our priorities, which in real life applications is not possible where processes come with different priorities. So we required a schedule which has near optimal turnaround time considering the priorities of the jobs.

If we neglect the priorities of the jobs and go for a execution sequence according to their CPU bursts, then we may end up with a set of processes such that the process with lowest burst time has the highest I/O burst and vice-versa, in such cases SJF schedule may not give optimal TAT so we have to consider both the priorities of the jobs in schedule as well as their burst times.

In our proposed algorithm, We have taken two schedules , once ordered according to SJF and and other according to their priorities, the weightage of priority in the optimal schedule is w_p i.e. new required schedule should contain w_p % of priority and $(1-w_p)$% of SJF in the solution.

Let required time,

RT = w_p * sch_priority + $(1-w_p)$*sch_sjf (1)

where,

sch_priority = Total Turnaround Time (TAT) according to priority scheduling.

sch_sjf = Total Turnaround Time (TAT) according to Shortest-Job-First (SJF) scheduling.

Require Time (RT) is the Total Turnaround Time (TAT) of some schedule and we have to find that schedule or a schedule whose TAT is nearby RT. This may be happen that there is no schedule which with RT, therefore we go for schedule whose TAT is nearby RT.

- **FITNESS FUNCTION**

  Our objective is to find a schedule with minimum tardiness.

  Tardiness =modulus [completion time – required time (RT)]. (2)

  Fitness value = tardiness value.

  Required time (RT) in (1).

3.3. Genetic Algorithm (GA)

Scheduling of jobs is a combinatorial problem which consumes exponential time but GA applied to many scheduling problems to solve them in polynomial time.

A basic GA consists of five components. These components are as follows: first, a random number generator; second, a fitness evaluation unit; last three are the genetic operators Selection (Reproduction), Crossover and Mutation operations.

The basic algorithm is summarized below:

1: Initialize Population

The initial population contain 50 chromosome representing schedule of N jobs, each chromosome contain priorities of N jobs and each priority is unique and priority assign to each job is randomly using poisson distribution.

```
2: Repeat
3:    Evaluation
4:    Selection
5:    Crossover
6:    Mutation
7: Until requirements are met
```

The main function in initialize population is **encoding scheme,** each element of the population is called Chromosome. A chromosome consists of genes, to represent a gene we required some encoding schemes. In our algorithm we use **value encoding scheme,** in which value is the priority of job, each genes of chromosome is a priority of job.

Flow-Chart of GA Algorithm

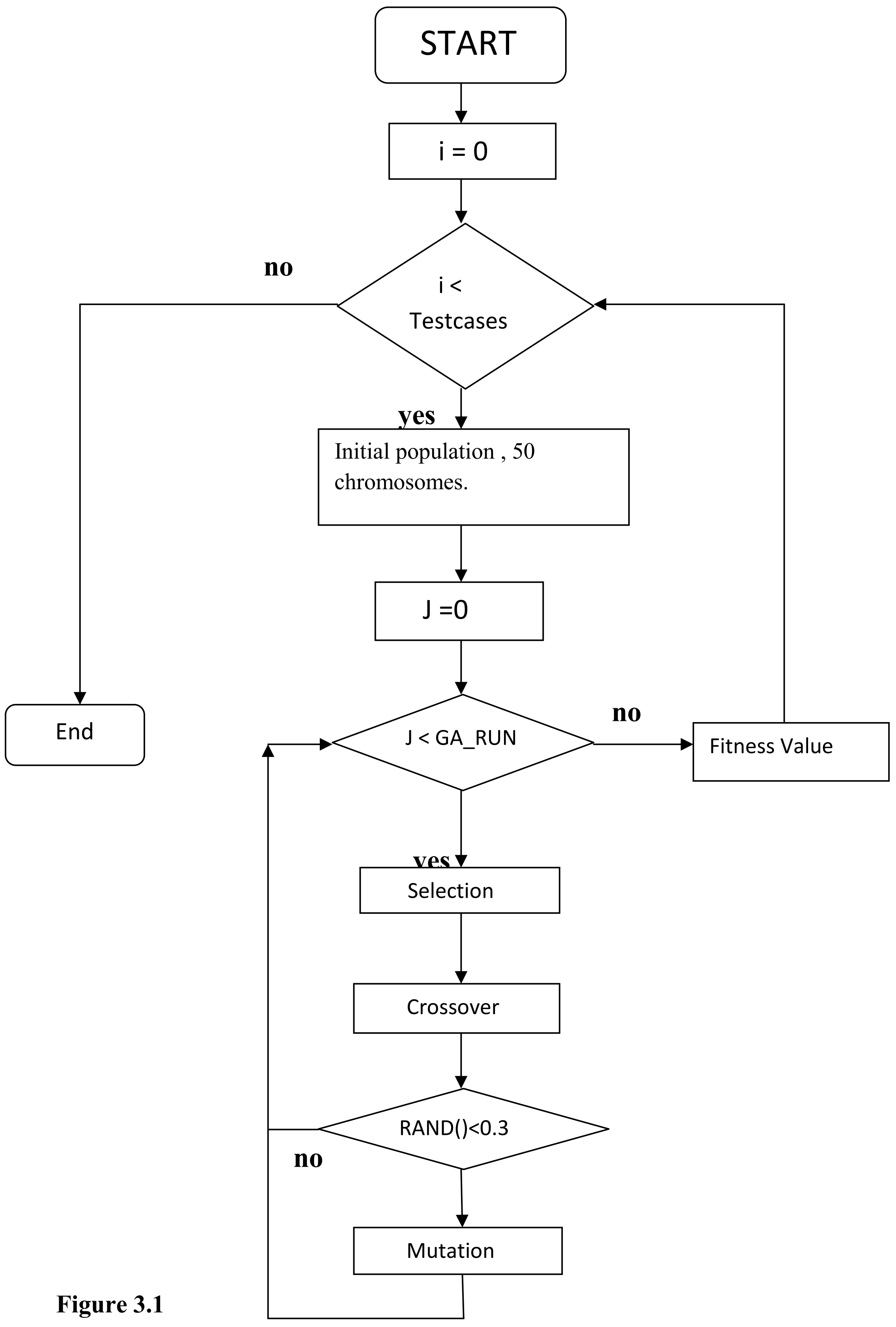

Figure 3.1

3.4. Modified-Genetic Algorithm (MGA)

The basic algorithm is summarized below:

1. Choosing an Encoding scheme.
2. Initialize population

   The initial population contain only two chromosome representing schedule of N jobs one ordered according to their priorities and the other according to the CPU bursts ,each chromosome contain priorities of N jobs and each priority is unique and priority assign to each job is randomly using poisson distribution.
3. **Repeat**
4. Evaluation

   Fitness function which is Tardiness (1).
5. Selection
6. Crossover

   There are several types of crossover:

   - One-point crossover.
   - Two-point crossover.
   - Half uniform crossover and Uniform crossover.
   - Ring crossover.

   In our algorithm we use Two-point crossover and Ring crossover.
7. Mutation

There are several types of mutation:

- Move
- Swap
- Move and Swap
- Rebalancing

In our algorithm we use swap.

Mutation is used to avoid local optimal.

8. **Until** requirements are met

Flow-Chart of Modified-GA Algorithm

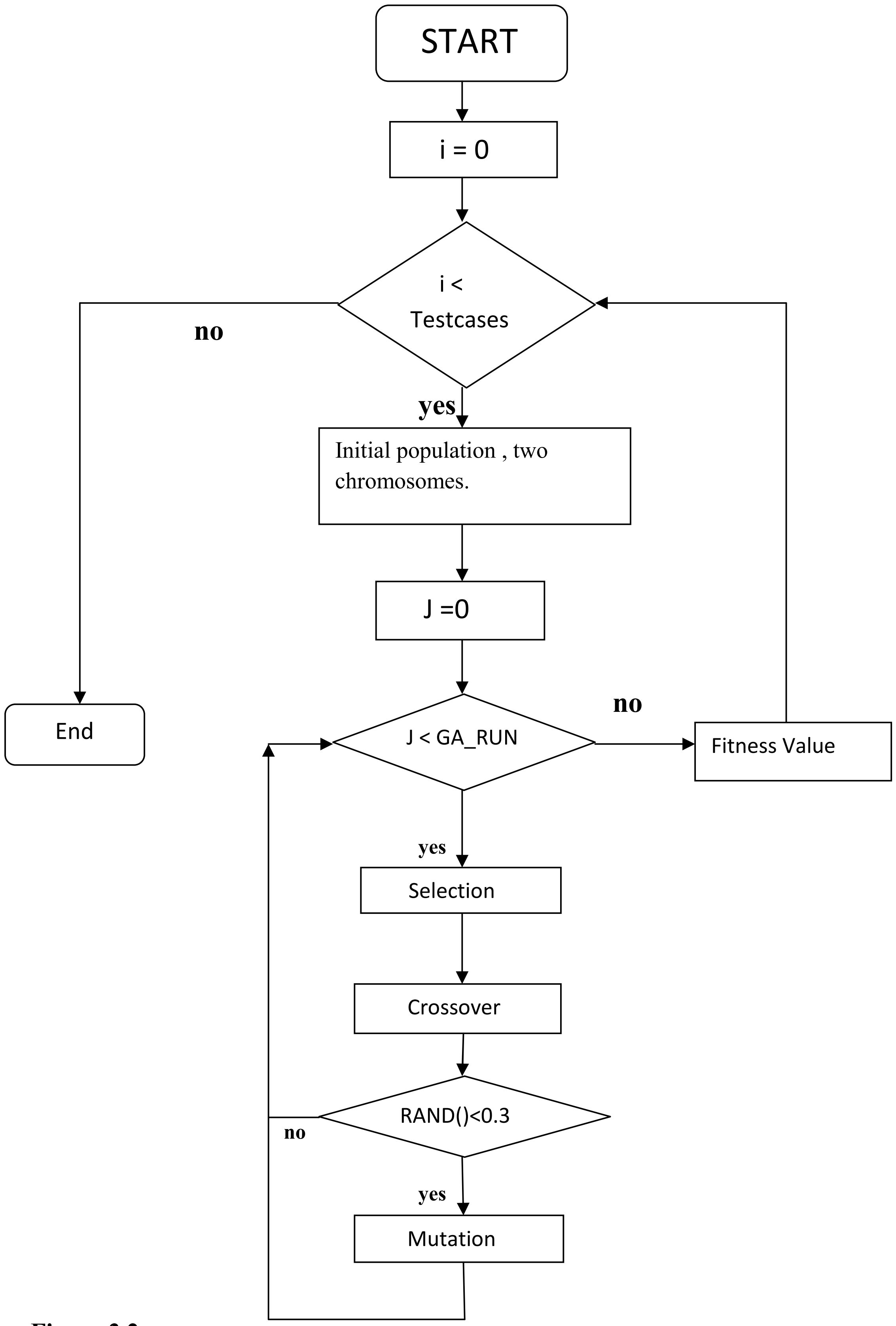

Figure 3.2

3.5. Artificial Bee Colony Algorithm (ABC)

In ABC algorithm, we imitate all these tasks in a computerized way. Employee starts the process of collecting nectars from different food sources. They bring back the nectars to the hive and exchange information about the food source they visited by a particular dance called waggle dance. Physical interpretation of waggle dance is, more a bee dances higher the quality of the food source visited by it. After seeing the waggle dance, onlooker bee decides the food source they are going to visit and this process continues until a food source is dead. When a food source is dead, it is abandoned, and scout bee starts searching for a new food source to replace the abandoned one.

In ABC algorithm, colony bees are divided into three groups: employed bees, onlooker bees and scout bees. The number of employees is equal to the number of food sources i.e. number of solutions in the population. The employee whose food source is exhausted is provided with a new food source by the scout bee. Each food source position is equivalent to a potential solution of optimization problem, with its quality or fitness value acting as the nectar amount. Whenever any bee finds a food source, it signals the other bees by stigmergy (waggle dance), the quality and the location of the food source. This attracts a large number of bees (onlooker bees) towards good food sources for further exploration.

The main steps of the algorithm are as below:

```
1: Initialize Population
2: **Repeat**
3: attach the employee bees to some food source( no. of food source equal to no. of employee bee)
4: attach the employee bees to some food source depending on their nectar amounts
5: Send the scouts bees for finding new food sources on the search area
6: Memorize the best food source found so far
7: **Until** requirements are met
```

The ABC algorithm has three operational phases:

- Scout bees do random search for the food, & find near optimal food sources, completely random.

- Employed bees visit food source and gather information about food source location and the quality. They have memory of the places they have visited before and quality of food there, & performs the local search to try to exploit the neighboring sources to locate the best.

Onlooker bees wait in the dance area to decide which food source is better on the basis of information provided by employees. They perform the global search for discovering the global optimum.

Pseudo code of ABC algorithm

```
1. Initialize the Control Parameters of the ABC Algorithm and randomly generate the
   priorities of jobs and each priority is unique in each chromosome, if in real life priorities
   are same then it is also manageable in our algorithm.
2. Set Iteration =0;
        Evaluate the nectar amount (fitness) of food sources (2);
3. Repeat Until (cycle is not equal to MAXCYCLE)
        for each employee bee do
        Search the neighbourhood of the food source for new solutions using equation. (3);
        $x_{ij}=y_{ij}+\psi(y_{kj}-y_{ij})$                                        (3)
        where j is the position of a randomly selected priority, $\varphi$ is a random number such that
        $\psi \epsilon[-1,1]$
        Check if $x_{ij}$ is within the bounds of the monitoring area
        Evaluate the fitness of the new food source using equation (2);
        Make a greedy selection between old solution and the new solution
4. Compute the probability $Prob_i$ of the solution using equation (4)
        $ProB_i = \frac{0.9*fitnessi}{fitnessbest}+0.1$                              (4)
5.For each onlooker bee do
        Generate a random number rand $\epsilon$ (0, 1) and select a food source depending on the
        value of ProB and rand;
        Look for new food source in the neighbourhood using the equation (3);
        Check if the new solution is within the bounds of the area
        Evaluate the fitness of the new food source using equation (3);
        Make a greedy selection between old solution and the new solution
```

```
6. Memorize the best solution found so far
7. Iterate through the trial array of food sources
if (the trials for a food source is greater than max trials)
replace the food source with a randomly generated food source;

Set trial =0
8. Cycle = Cycle +1;
```

3.6. Modified-Artificial Bee Colony Algorithm (MABC)

The main steps of the algorithm are as below :

```
1: Initialize Population
2: Repeat
3: attach the employee bees to some food source( no. of food source equal to no. of employee bee)
4: attach the employee bees to some food source depending on their nectar amounts
5: Send the scouts bees for finding new food sources on the search area
6: Crossover operation
7: Memorize the best food source found so far
8: Until requirements are met
```

In MABC algorithm, colony bees are divided into four groups: employed bees, onlooker bees, scout bees and crossover operation. The number of employees is equal to the number of food sources i.e. number of solutions in the population. The employee whose food source is exhausted is provided with a new food source by the scout bee. Each food source position is equivalent to a potential solution of optimization problem, with its quality or fitness value acting as the nectar amount. Whenever any bee finds a food source, it signals the other bees by stigmergy (waggle dance), the quality and the location of the food source. This attracts a large number of bees (onlooker bees) towards good food sources for further exploration. In crossover operation, generate parent population of food source by applying tournament selection operation in which for each tournament fitness value select three fitness value from total population randomly and the best of three fitness is the fitness of tournament. Select a certain amount of worse food sources based on the crossover probability value; for each selected food source in tournament apply two point crossover and ring crossover, the best fitness of these two crossover is the fitness of crossover applied food source.

Pseudo code of Modified-ABC algorithm is summarized below:

```
1. Initialize the Control Parameters of the ABC Algorithm and randomly generate the
   priorities of jobs and each priority is unique in each chromosome, if in real life priorities
   are same then it is also manageable in our algorithm.
2. Set Iteration =0;
      Evaluate the nectar amount (fitness) of food sources (2);
3. Repeat Until (cycle is not equal to MAXCYCLE)
      for each employee bee do
      Search the neighbourhood of the food source for new solutions using equation. (3);
      $x_{ij}=y_{ij}+\psi(y_{kj}-y_{ij})$                                   (3)
      where j is the position of a randomly selected priority, $\varphi$ is a random number such that
       $\psi \in [-1,1]$
       Check if $x_{ij}$ is within the bounds of the monitoring area
       Evaluate the fitness of the new food source using equation (2);
       Make a greedy selection between old solution and the new solution
4. Compute the probability $Prob_i$ of the solution using equation (4)
      $ProB_i = \frac{0.9*fitnessi}{fitnessbest}+0.1$                     (4)
5.For each onlooker bee do
      Generate a random number rand $\in$ (0, 1) and select a food source depending on the
      value of ProB and rand;
      Look for new food source in the neighbourhood using the equation (3);
      Check if the new solution is within the bounds of the area
      Evaluate the fitness of the new food source using equation (3);
      Make a greedy selection between old solution and the new solution
6. Crossover Operation
     Generate the parent population of food source by applying tournament selection;
Select a certain amount of worse food sources based on the crossover probability value;
for each selected food source do
Select two parents randomly from the parent population
```

```
Produce two new food sources by crossing the selected parents;
Apply greedy selection for the selected food source and the newly produced food sources;

7. Memorize the best solution found so far
8. Iterate through the trial array of food sources
if (the trials for a food source is greater than max trials)
replace the food source with a randomly generated food source;

Set trial =0
9. Cycle = Cycle +1;
```

CHAPTER 4

SIMULATION RESULTS & ANALYSIS

Simulation is considered as flexible and efficient tool to evaluate the performance of the algorithm working under vivid environmental conditions. In this chapter, MGA and MABC technique proposed in chapter 4 are evaluated on a simulation platform. The pre-existing algorithms GA and ABC are also evaluated on a simulation platform. The performance of the proposed algorithm is compared with other pre-existing algorithm in terms of number of iteration, number of test case, requirement percentage and tardiness (fitness value).

4.1 Simulation Setup

Microsoft Visual C++ is the tool we used for simulation and performance evaluation of GA, MGA, ABC and MABC technique. Our aim with the simulation is to compare GA and ABC with our MGA and MABC algorithm in respect to number of iteration, number of test case, requirement percentage and tardiness (fitness value).
For the experiment we have randomly generated an input set using Poisson distribution with some factors which are taken for experiment are shown in table 1, number of test cases are 20 with random number of processes respectively. The algorithm is implemented in C with system having 4GB ram, i5 2.6GHz.

	MEAN	VARIANCE	RANGE
Process	50	30	[20-80]
Arrival	5	5	[0-10]
Burst	20	19	[1-39]
Priority	25	24	[1-49]

Table 4.1

Requirement percentage is varies from 0.05 to 0.90, 0 and 1 is not include because 0 indicate require sequence is the sequence of SJF scheduling which we already have and 1 indicate require sequence is the sequence of Priority scheduling which we already have.

Requirement Percentage																		
0.05	0.1	0.15	0.2	0.25	0.3	0.35	0.4	0.45	0.5	0.55	0.6	0.65	0.7	0.75	0.8	0.85	0.9	0.95

Table 4.2

In GA, the number of chromosomes is 50, the mutation probability is 0.03.

In MGA, the number of chromosomes is 2 i.e. one according to SJF and other according to Priority, the mutation probability is 0.03.

In ABC and MABC, the values of the parameters of the probabilistic model are as follows: γ_1=1, γ_2=0, β_1=1, β_2=0.5.

The crossover probability is 0.2 and the tournament population size is 30.

For all algorithms maximum number of cycle (MAXCYCLE): 1000, number of test cases is 20.

Number of iteration indicates solution converse after such number of iterations.

Total number of food source in ABC and MABC are 50, each food source has priorities as parameter and the range of priority is from 1 to number of processes in particular test case.

4.2 Performance Evaluation- GA and MGA

With the above parameters mention in section 4.1 the performance evaluation of between GA and MGA is discussed in this section. The performance of the proposed MGA algorithm is compared with pre-existing GA algorithm in terms of number of iteration, number of test case, requirement percentage and tardiness (fitness value).

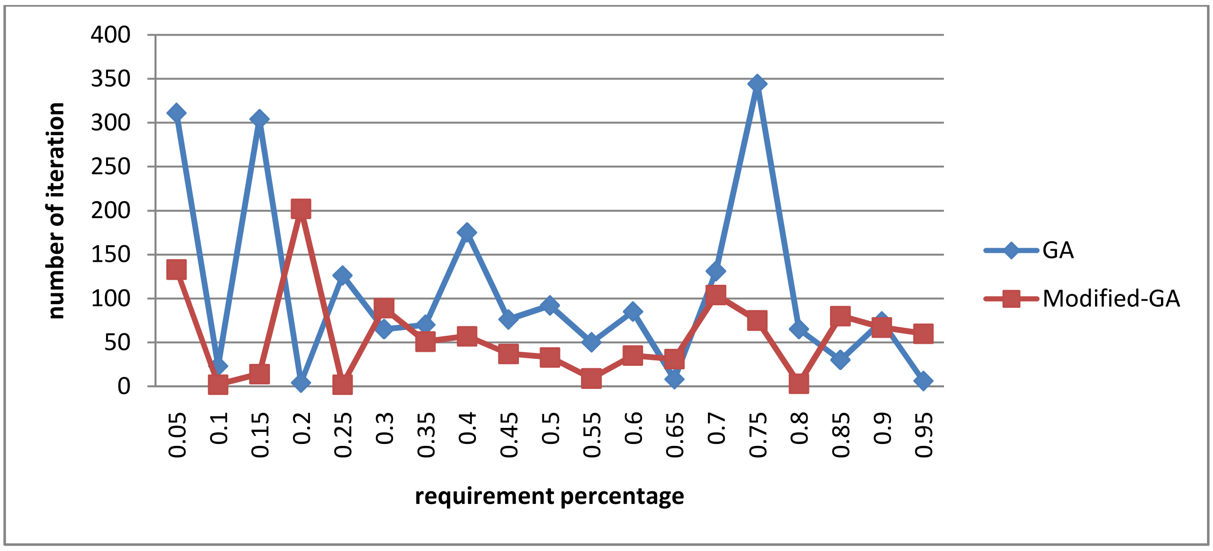

Figure 4.1

In above figure, a test case which have 65 number of processes, graph represents the number of iteration in which solution (tardiness) converse with the varies of requirement percentage.

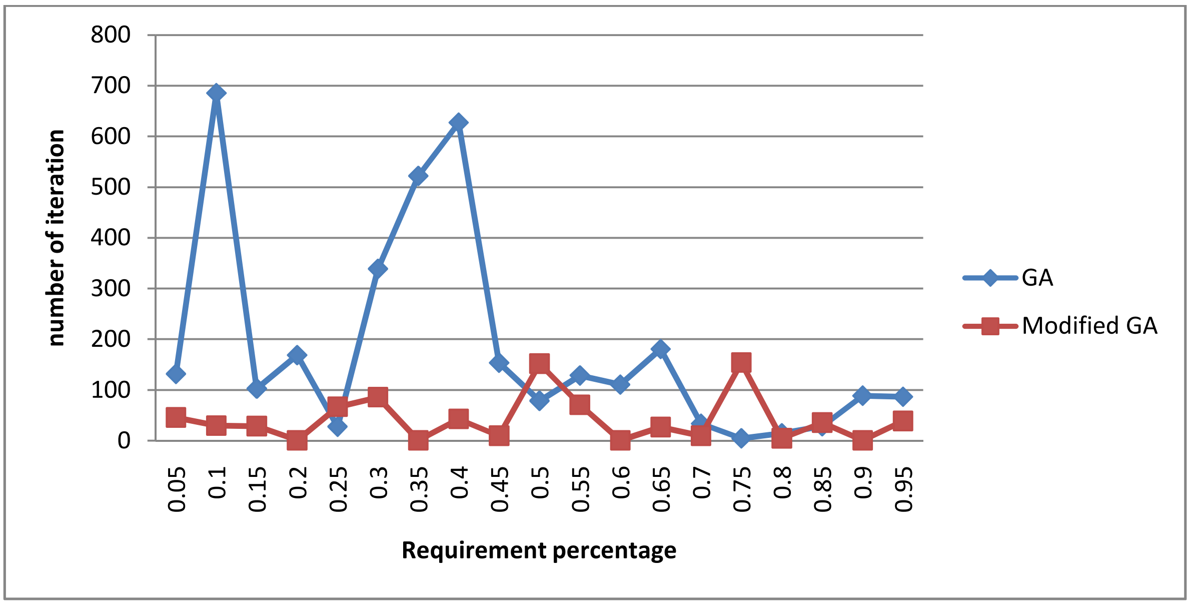

Figure 4.2

In above figure, a test case which have 79 number of processes, graph represents the number of iteration in which solution (tardiness) converse with the varies of requirement percentage.

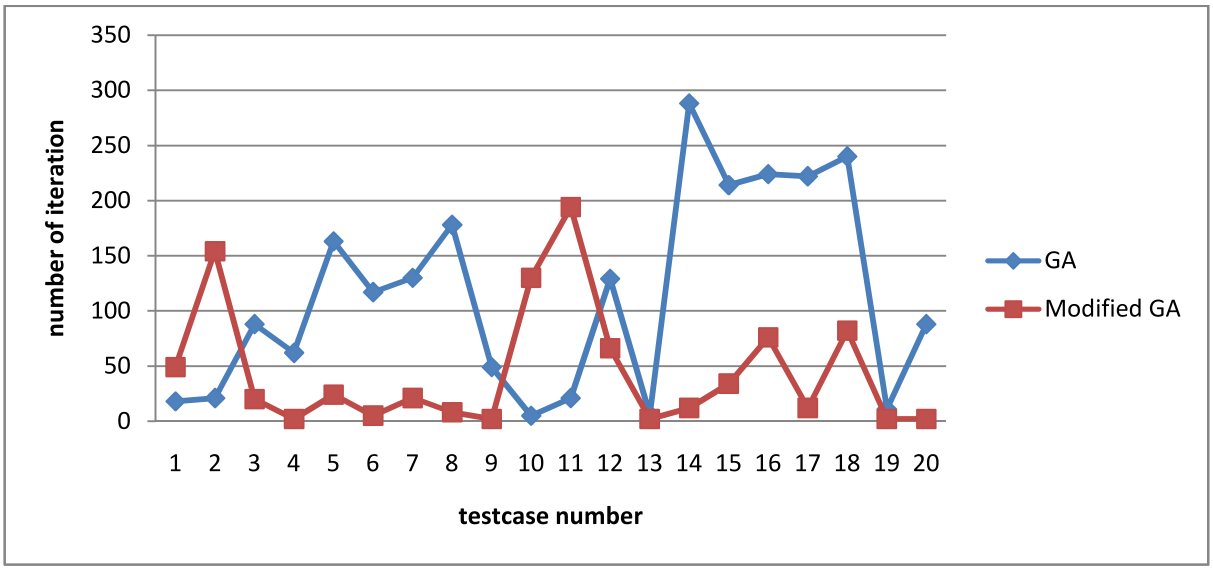

Figure 4.3

In above figure, requirement percentage is 0.3; graph represents the number of iteration in which solution (tardiness) converse with the different number of test cases.

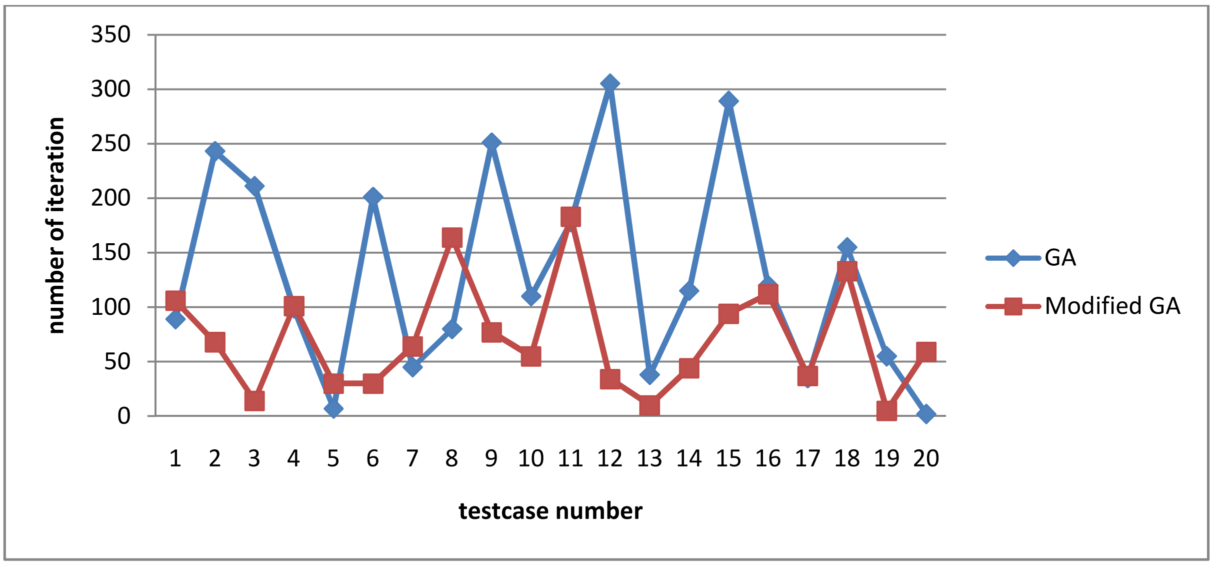

Figure 4.4

In above figure, requirement percentage is 0.5; graph represents the number of iteration in which solution (tardiness) converse with the different number of test cases.

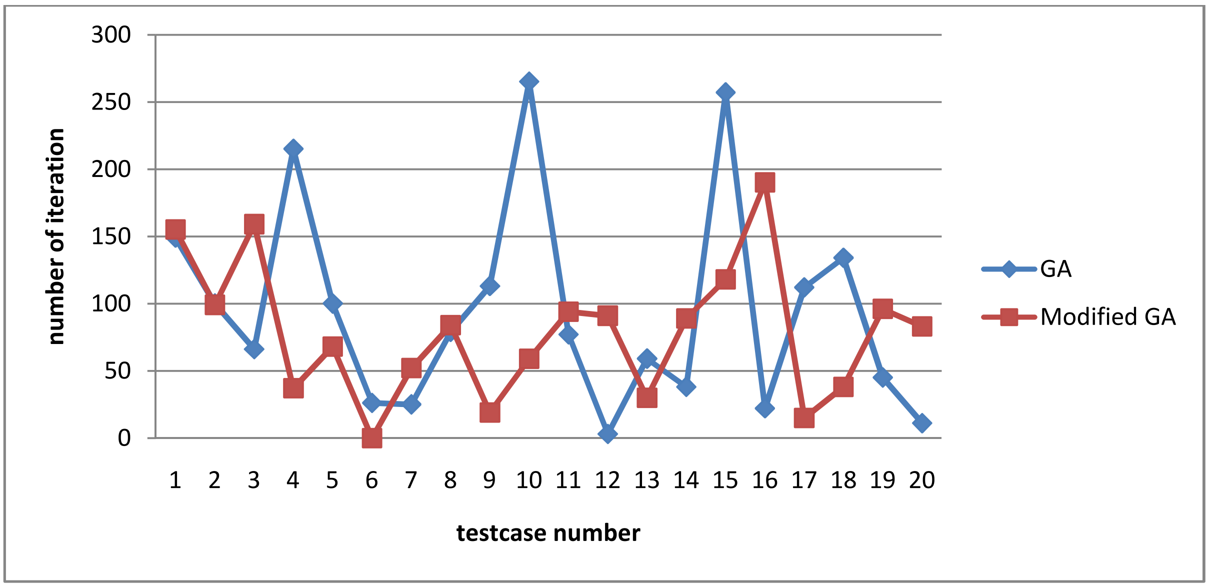

Figure 4.5

In above figure, requirement percentage is 0.7; graph represents the number of iteration in which solution (tardiness) converse with the different number of test cases.

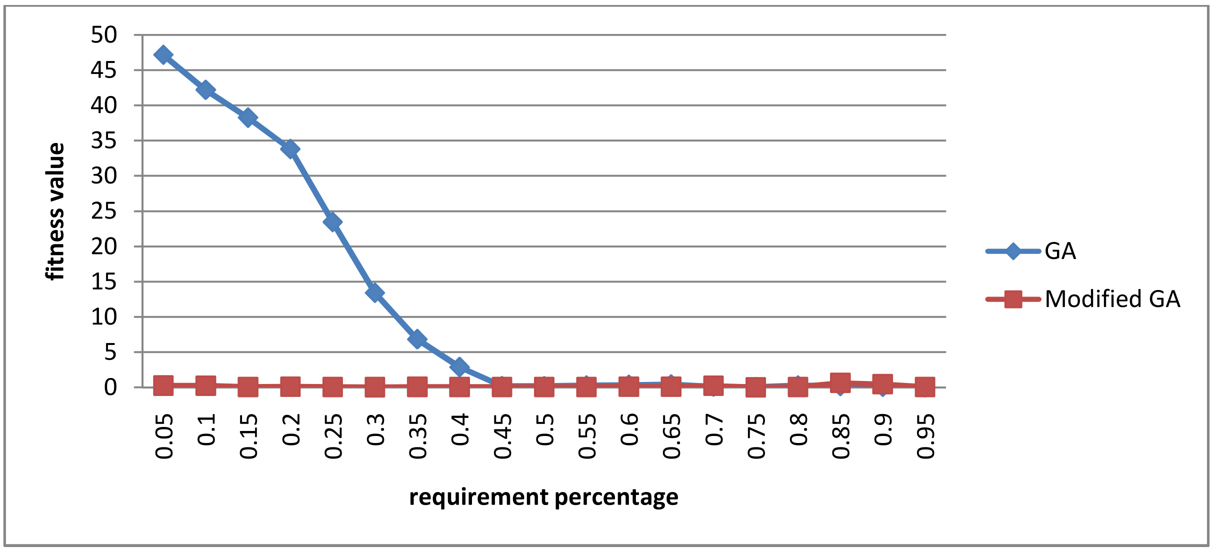

Figure 4.6

In above figure, a test case which have 65 number of processes, graph represents the solution (tardiness / fitness value) converse with the varies of requirement percentage.

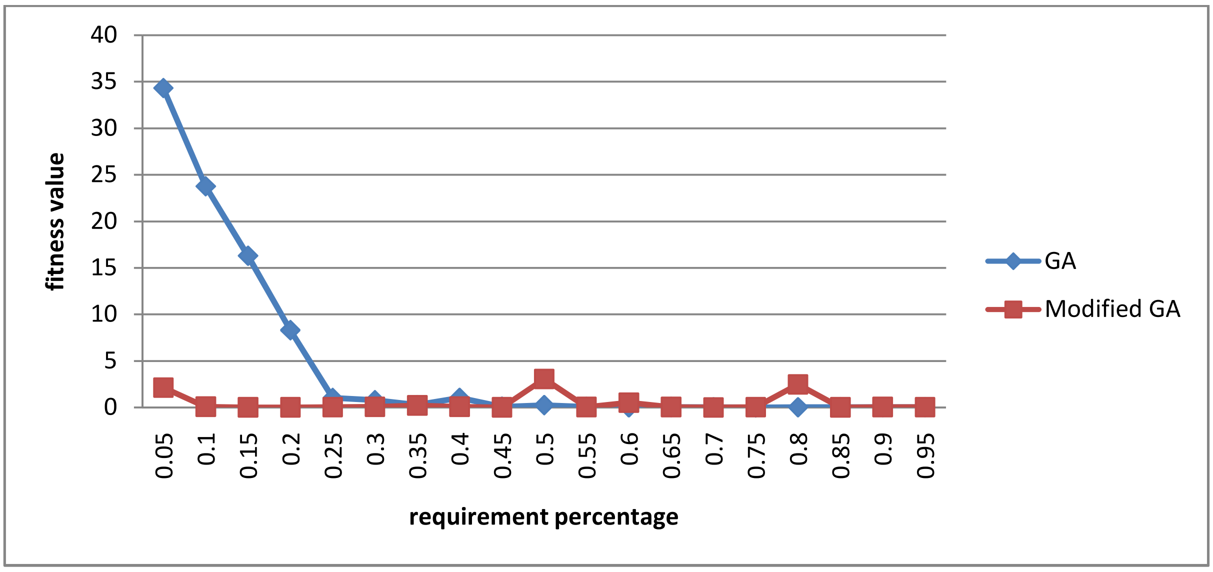

Figure 4.7

In above figure, a test case which have 79 number of processes, graph represents the solution (tardiness / fitness value) converse with the varies of requirement percentage.

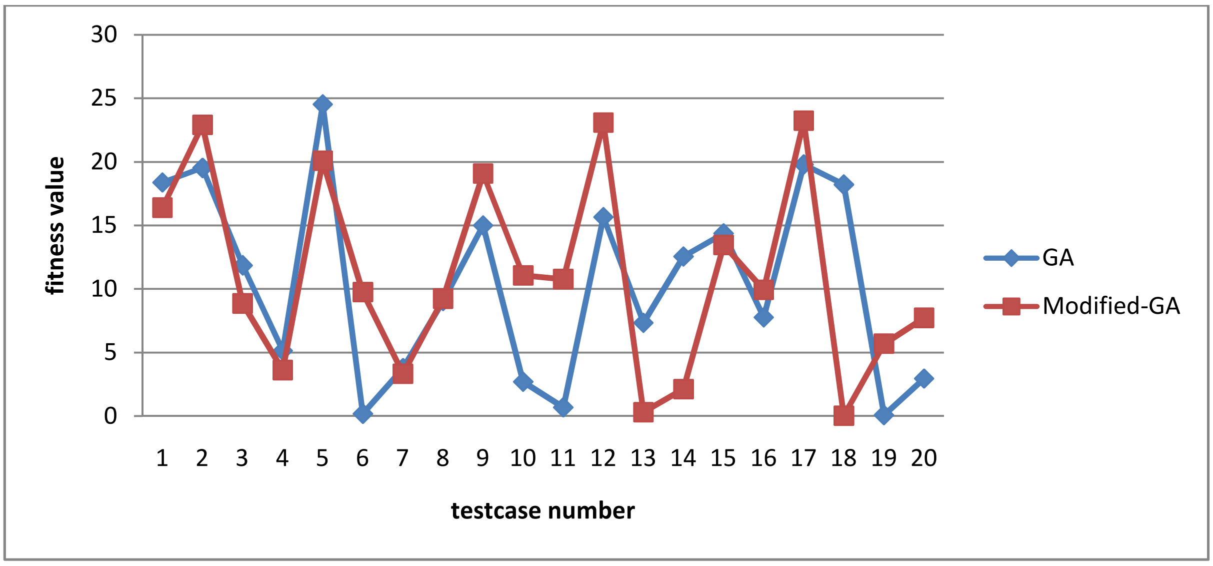

Figure 4.8

In above figure, requirement percentage is 0.3; graph represents the solution (tardiness / fitness value) converse with the different number of test cases.

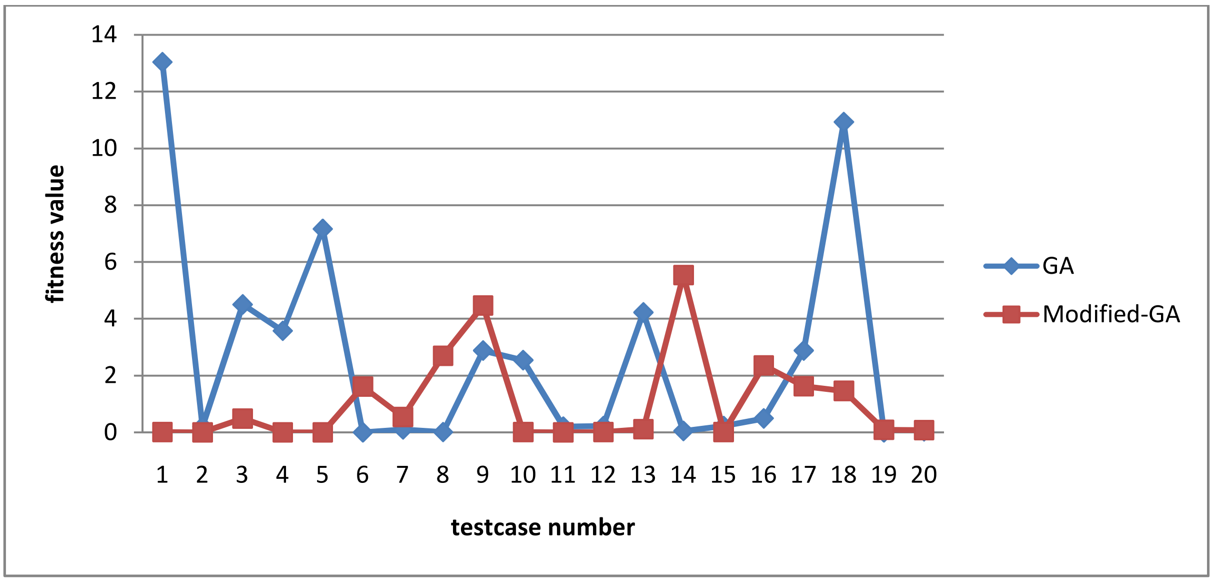

Figure 4.9

In above figure, requirement percentage is 0.5; graph represents the solution (tardiness / fitness value) converse with the different number of test cases.

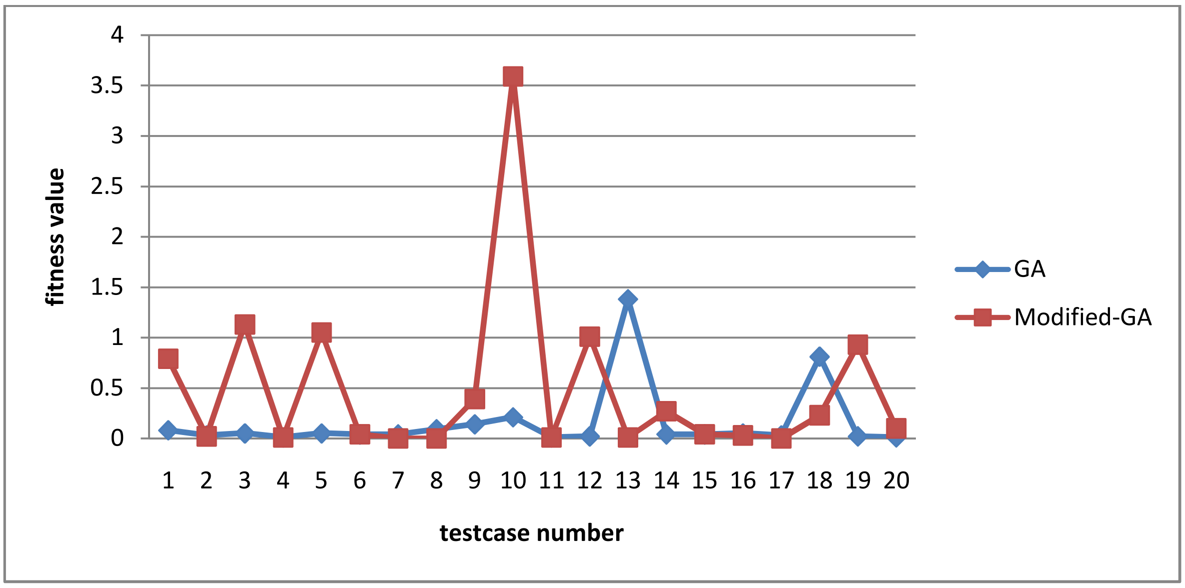

Figure 4.10

In above figure, requirement percentage is 0.7; graph represents the solution (tardiness / fitness value) converse with the different number of test cases.

4.3 Performance Evaluation- ABC and MABC

With the above parameters mention in section 4.1 the performance evaluation of between ABC and MABC is discussed in this section. The performance of the proposed MABC algorithm is compared with pre-existing ABC algorithm in terms of number of iteration, number of test case, requirement percentage and tardiness (fitness value).

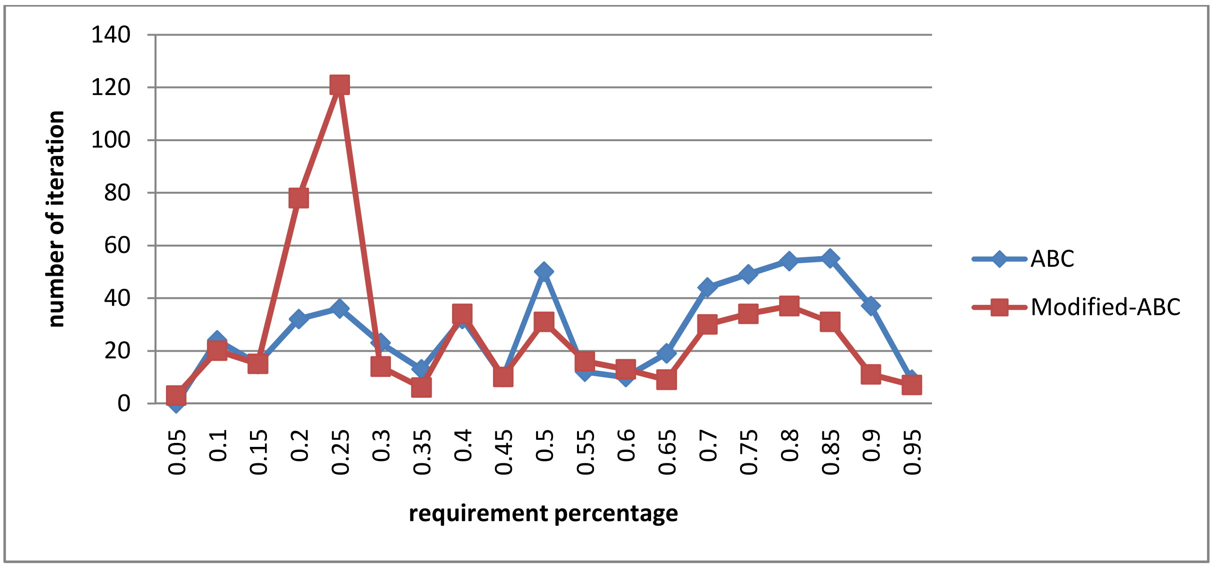

Figure 4.11

In above figure, a test case which have 65 number of processes, graph represents the number of iteration in which solution (tardiness) converse with the varies of requirement percentage.

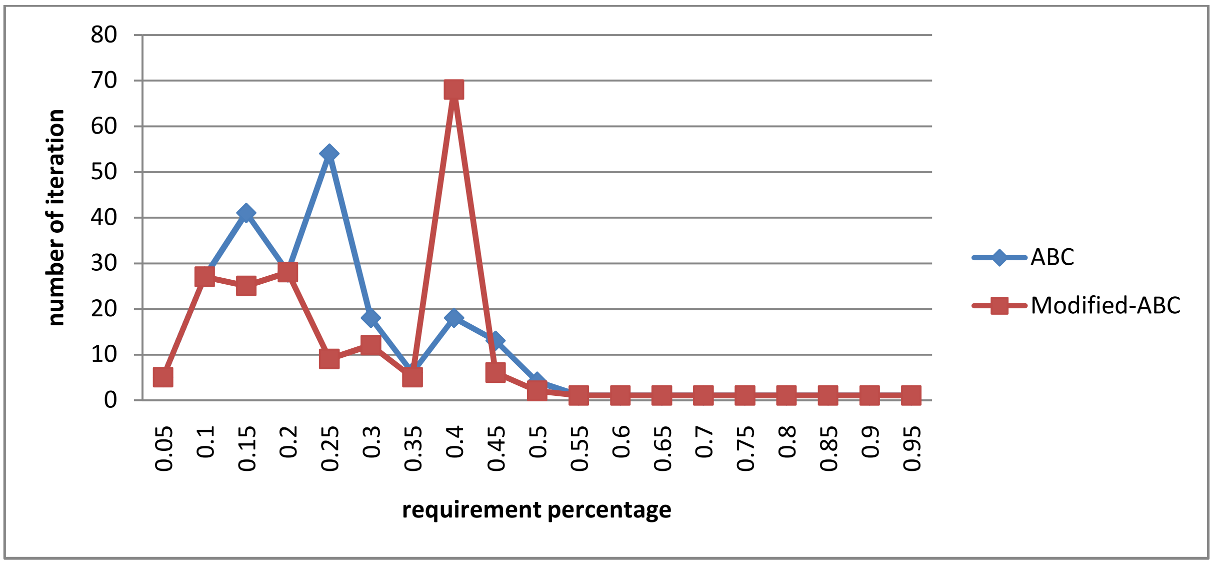

Figure 4.12

In above figure, a test case which have 79 number of processes, graph represents the number of iteration in which solution (tardiness) converse with the varies of requirement percentage.

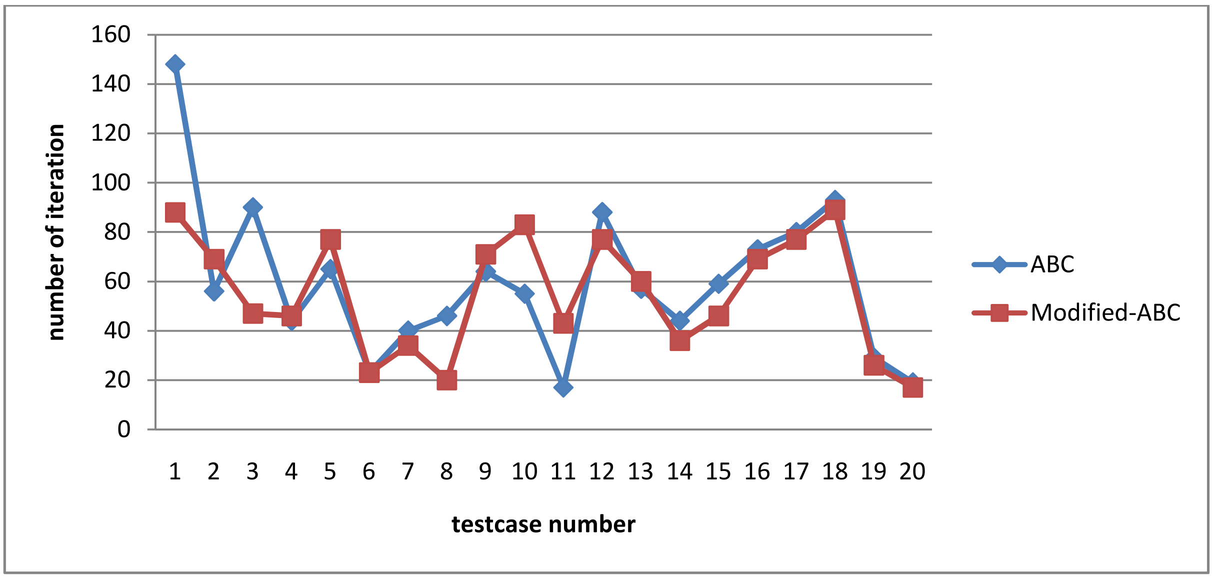

Figure 4.13

In above figure, requirement percentage is 0.3; graph represents the number of iteration in which solution (tardiness) converse with the different number of test cases.

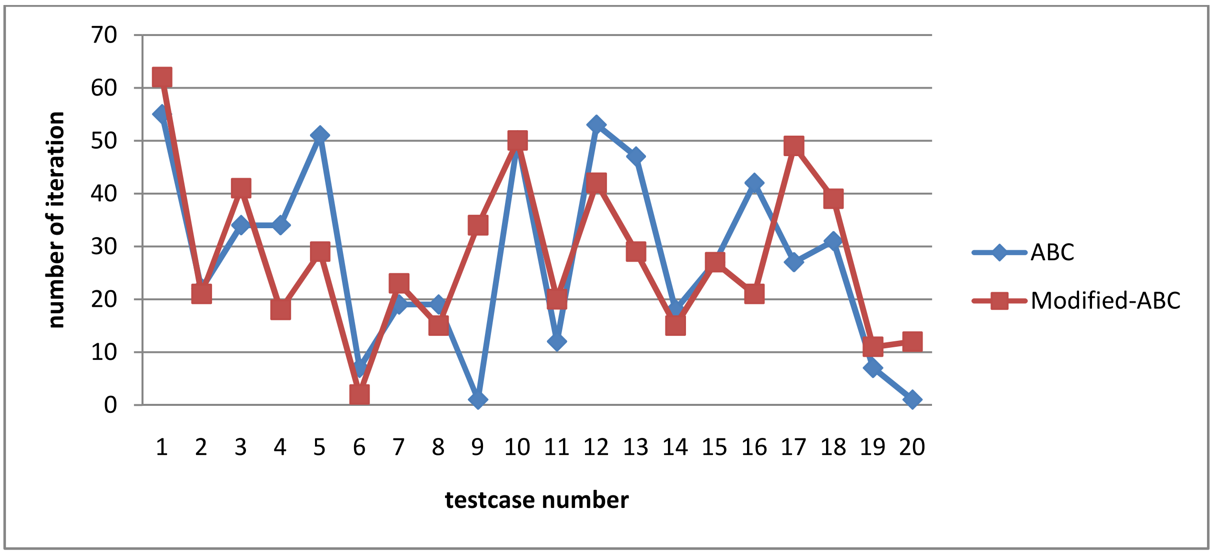

Figure 4.14

In above figure, requirement percentage is 0.5; graph represents the number of iteration in which solution (tardiness) converse with the different number of test cases.

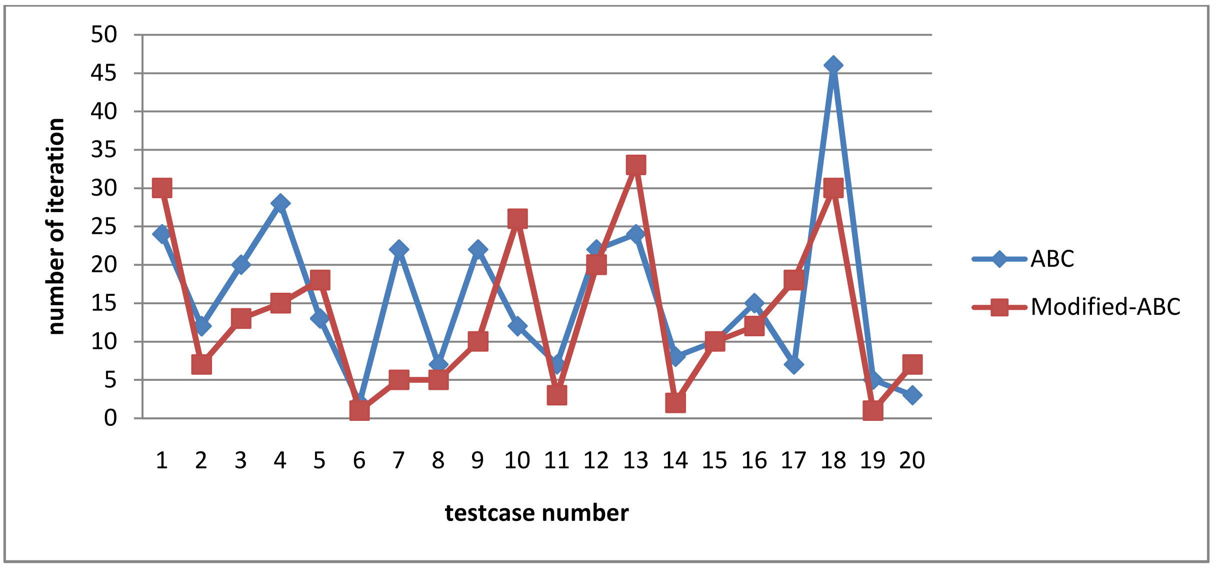

Figure 4.15

In above figure, requirement percentage is 0.7; graph represents the number of iteration in which solution (tardiness) converse with the different number of test cases.

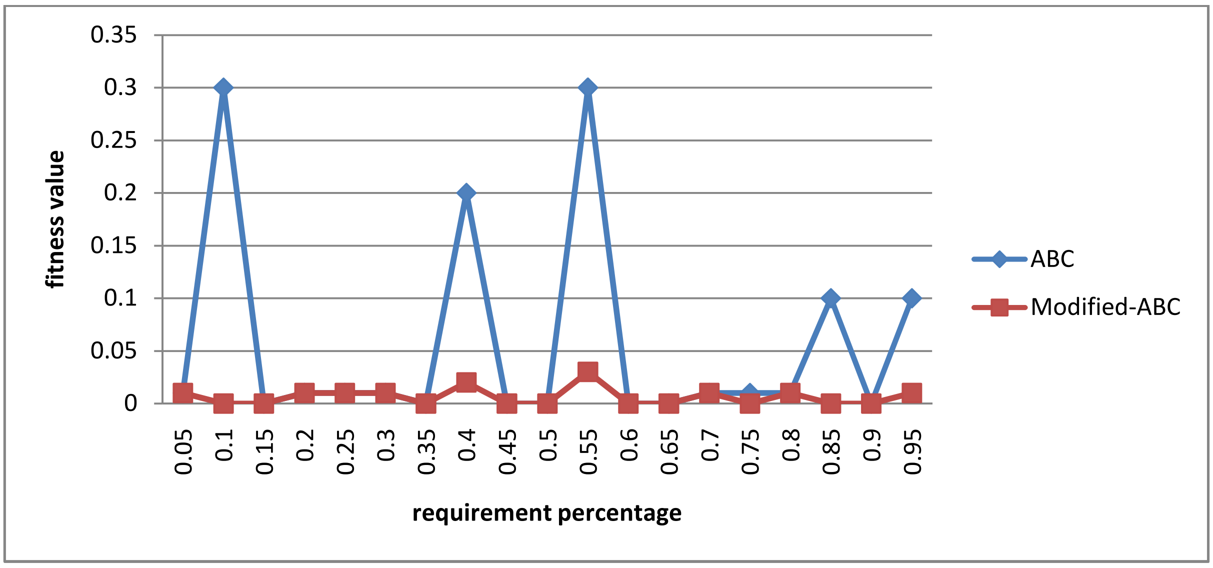

Figure 4.16

In above figure, a test case which have 65 number of processes, graph represents the solution (tardiness / fitness value) converse with the varies of requirement percentage.

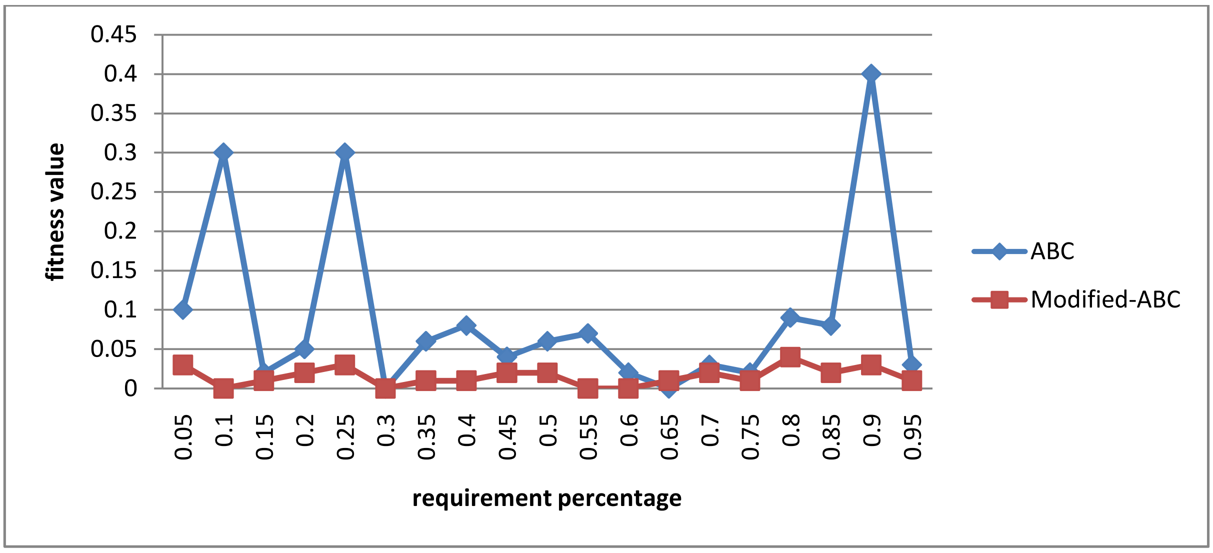

Figure 4.17

In above figure, a test case which have 79 number of processes, graph represents the solution (tardiness / fitness value) converse with the varies of requirement percentage.

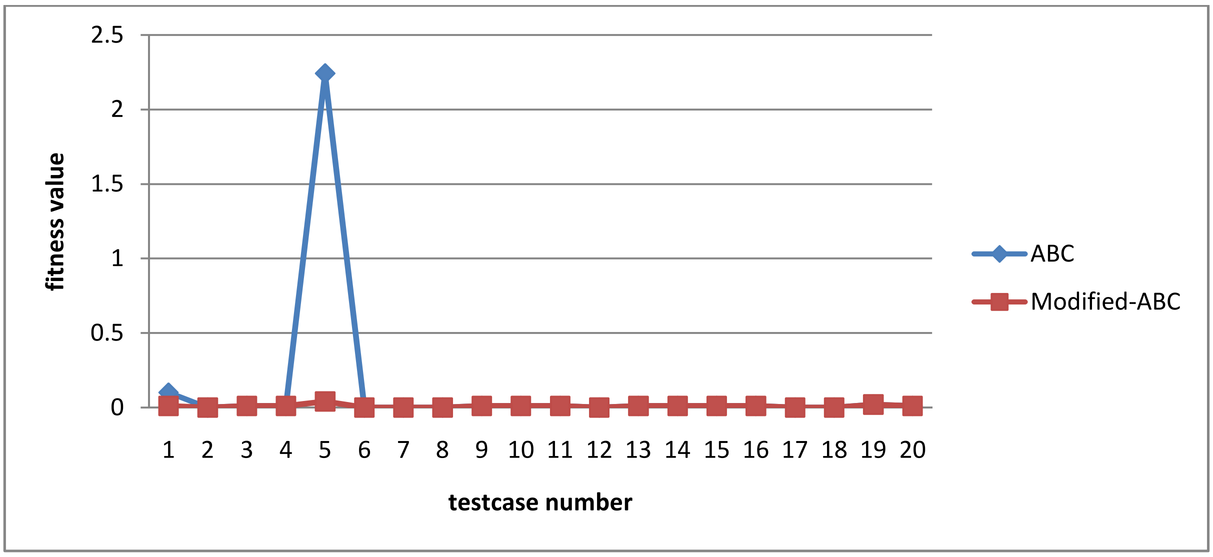

Figure 4.18

In above figure, requirement percentage is 0.3; graph represents the solution (tardiness / fitness value) converse with the different number of test cases.

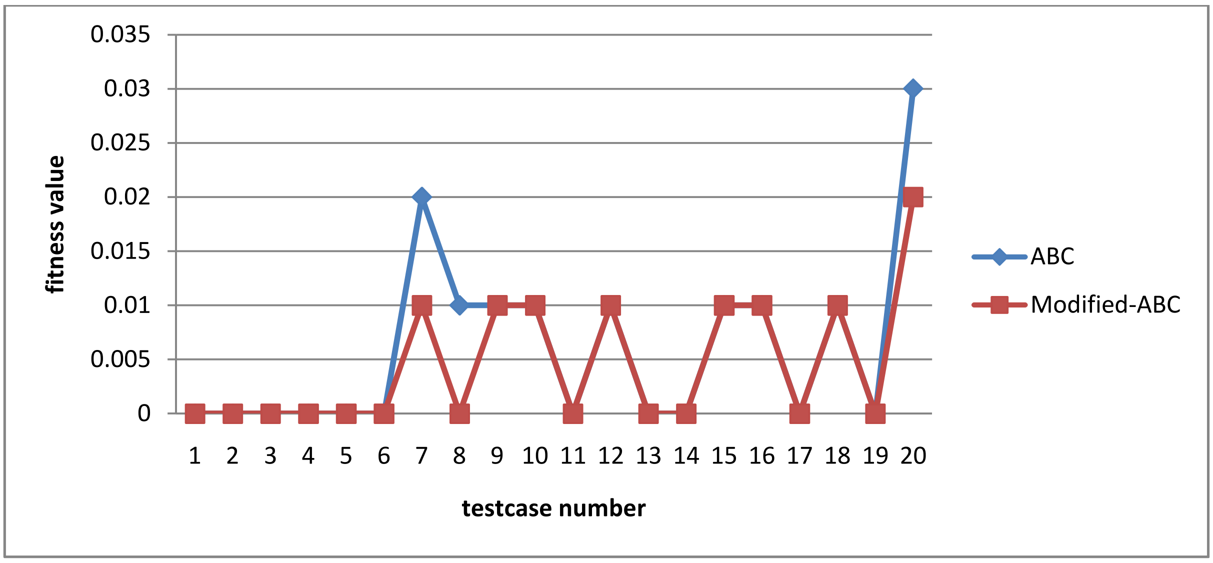

Figure 4.19

In above figure, requirement percentage is 0.5; graph represents the solution (tardiness / fitness value) converse with the different number of test cases.

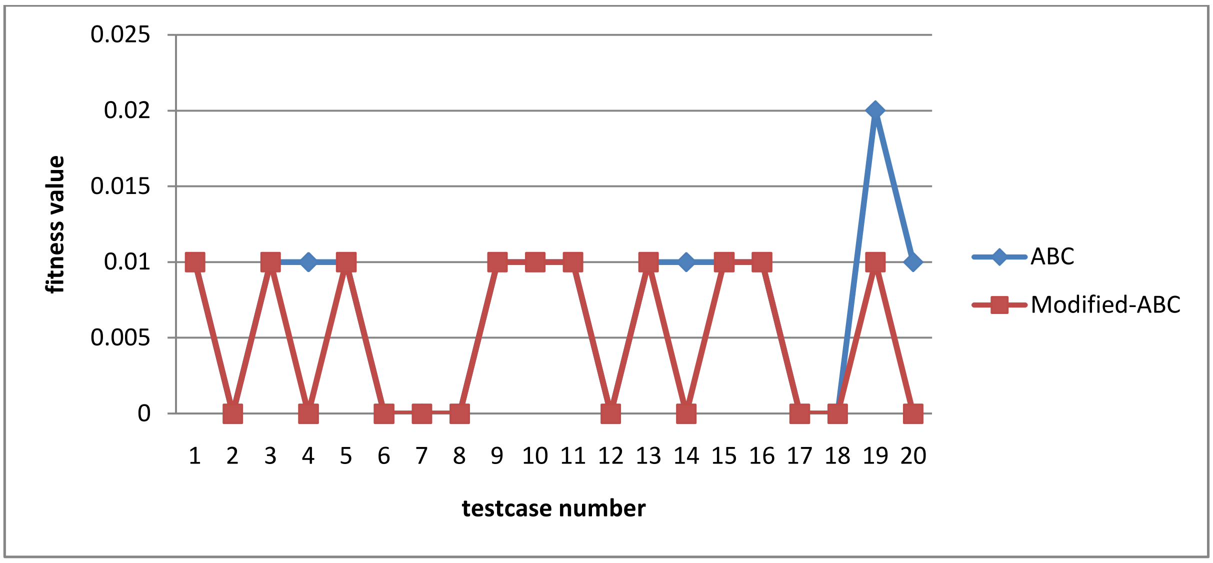

Figure 4.20

In above figure, requirement percentage is 0.7; graph represents the solution (tardiness / fitness value) converse with the different number of test cases.

4.3 Performance Evaluation- GA, MGA, ABC and MABC

With the above parameters mention in section 4.1 the performance evaluation of between GA, MGA, ABC and MABC is discussed in this section. The performance of the proposed

MGA, MABC algorithm is compared with pre-existing GA, ABC algorithm in terms of number of iteration, number of test case, requirement percentage and tardiness (fitness value).

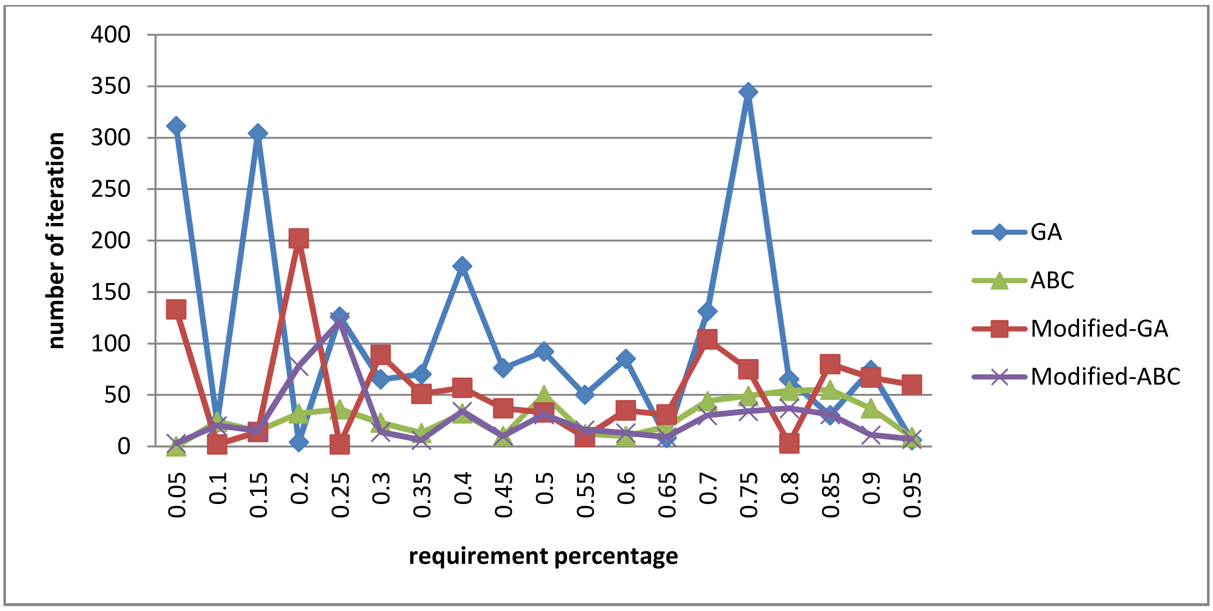

Figure 4.21

In above figure, a test case which have 65 number of processes, graph represents the number of iteration in which solution (tardiness) converse with the varies of requirement percentage.

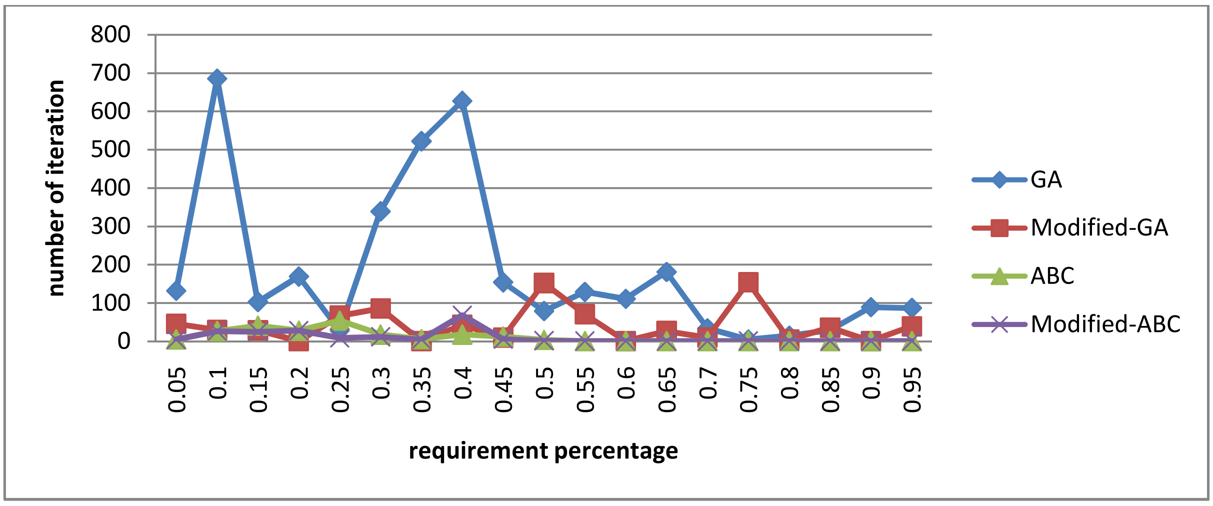

Figure 4.22

In above figure, a test case which have 65 number of processes, graph represents the number of iteration in which solution (tardiness) converse with the varies of requirement percentage.

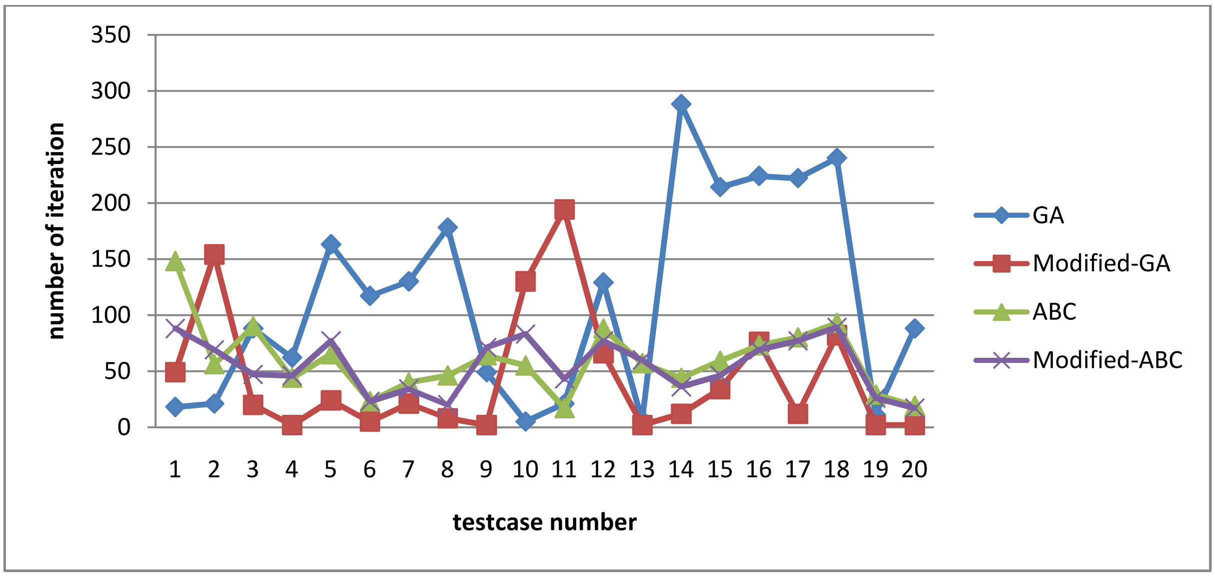

Figure 4.23

In above figure, requirement percentage is 0.3; graph represents the number of iteration in which solution (tardiness) converse with the different number of test cases.

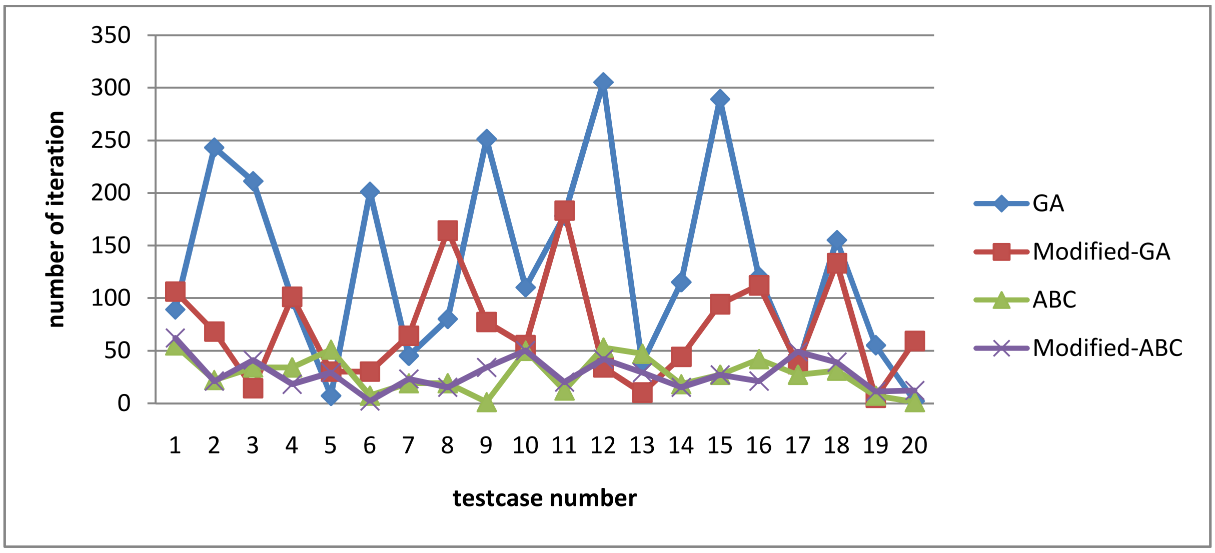

Figure 4.24

In above figure, requirement percentage is 0.5; graph represents the number of iteration in which solution (tardiness) converse with the different number of test cases.

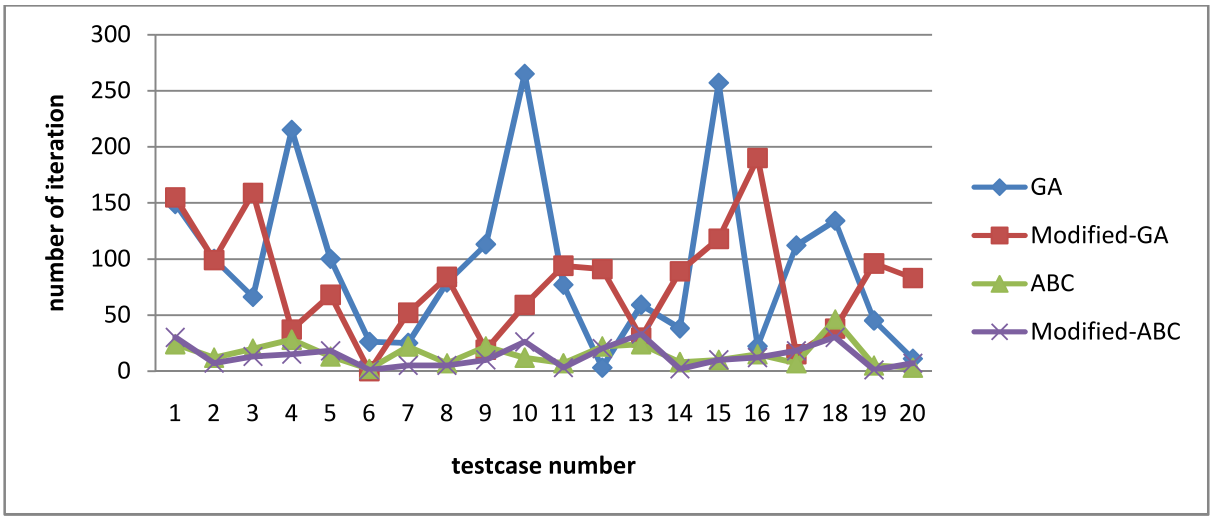

Figure 4.25

In above figure, requirement percentage is 0.7; graph represents the number of iteration in which solution (tardiness) converse with the different number of test cases.

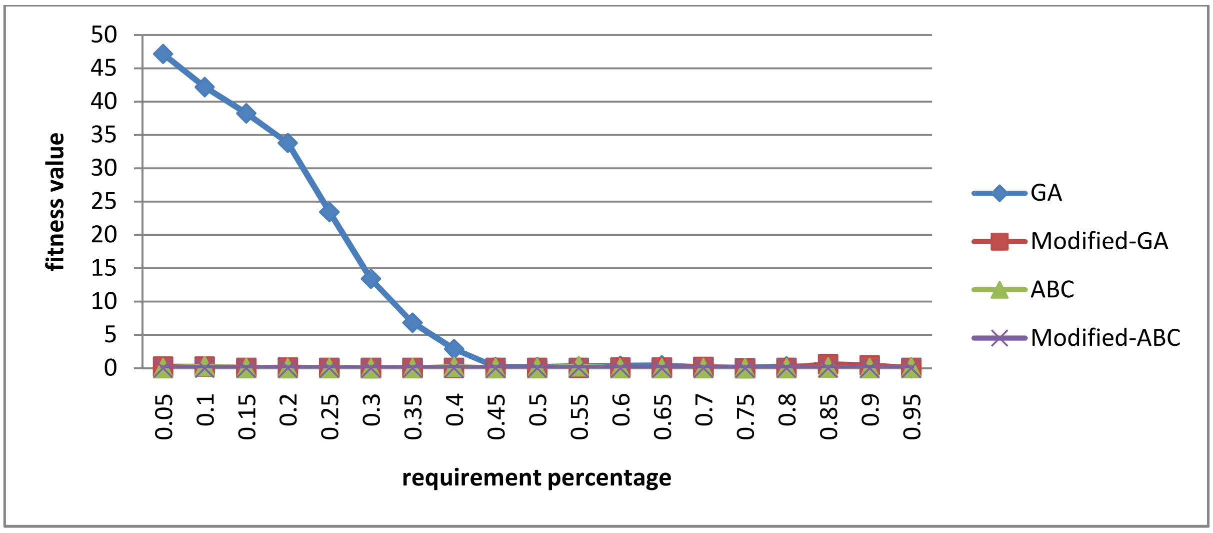

Figure 4.26

In above figure, a test case which have 65 number of processes, graph represents the solution (tardiness / fitness value) converse with the varies of requirement percentage.

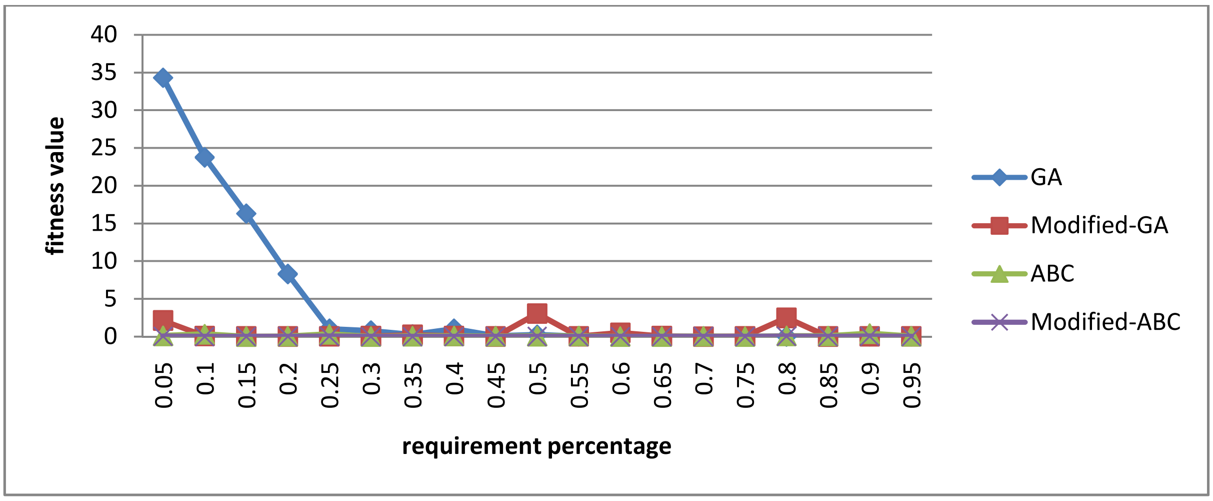

Figure 4.27

In above figure, a test case which have 79 number of processes, graph represents the solution (tardiness / fitness value) converse with the varies of requirement percentage.

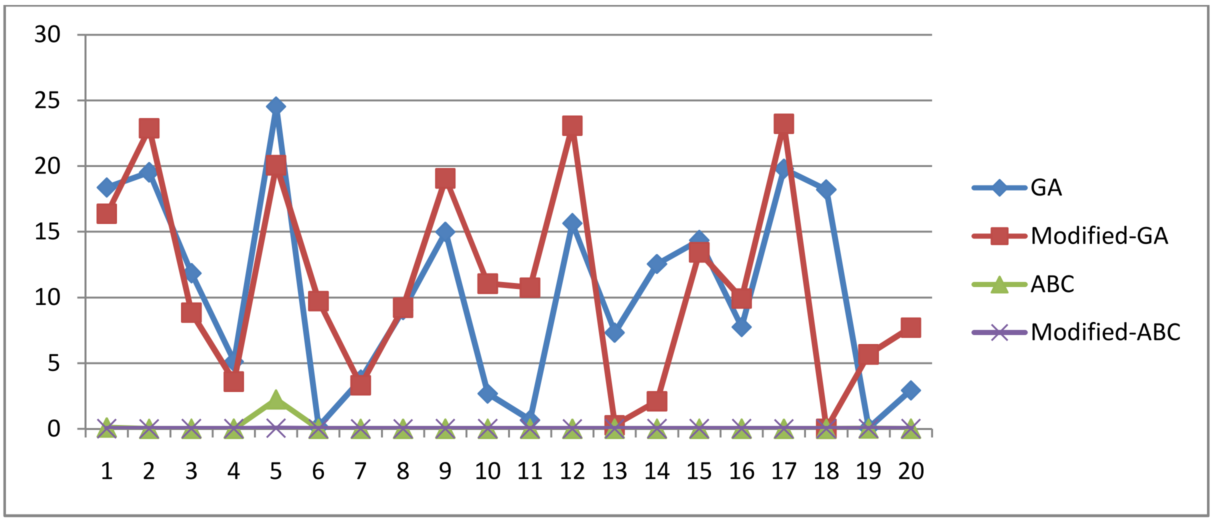

Figure 4.28

In above figure, requirement percentage is 0.3; graph represents the solution (tardiness / fitness value) converse with the different number of test cases.

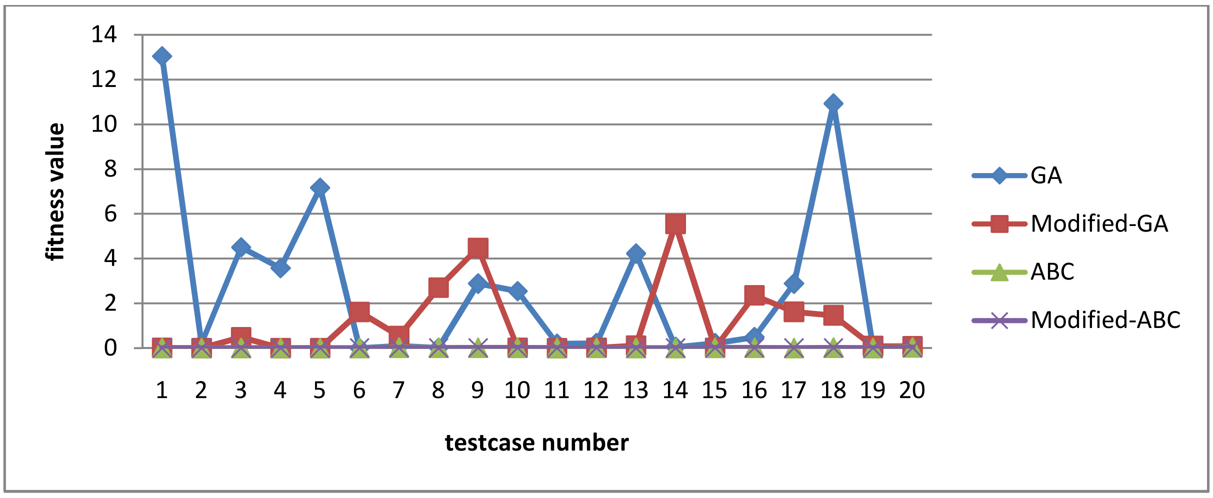

Figure 4.29

In above figure, requirement percentage is 0.5; graph represents the solution (tardiness / fitness value) converse with the different number of test cases.

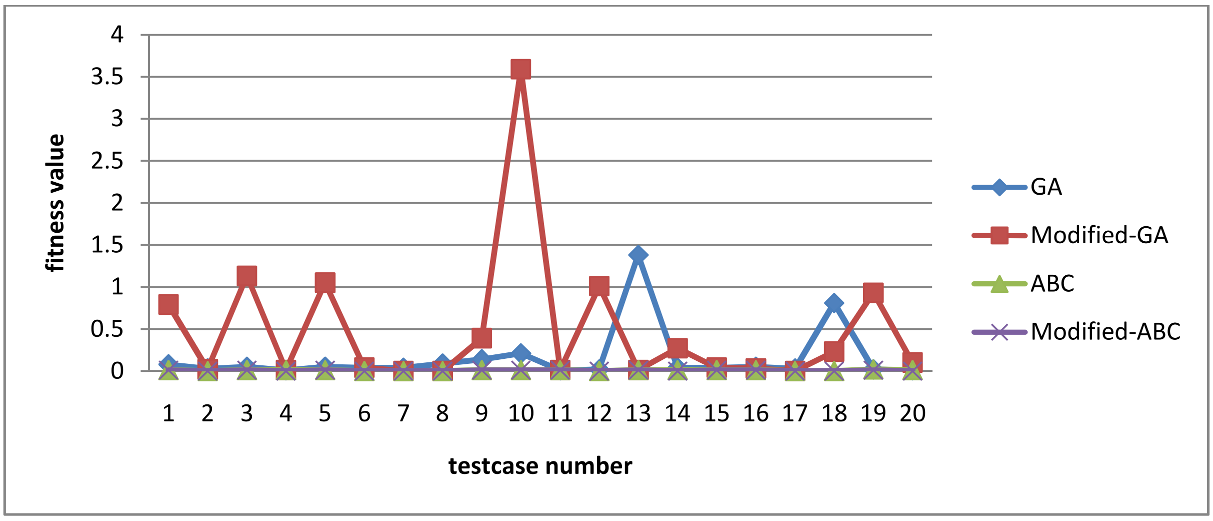

Figure 4.30

In above figure, requirement percentage is 0.7; graph represents the solution (tardiness / fitness value) converse with the different number of test cases.

CHAPTER 5

CONCLUSION AND FUTURE WORK

Process scheduling problem are a combinatorial problem in which selection and arrangement of jobs are key factor. Different algorithm and techniques are developed to find above factor, there are several approximation techniques which provide nearby desirable optimal solution. In our research a deadline aware approximation algorithm is applied (i.e. Genetic Algorithm, Artificial Bee Colony and their modification) for scheduling jobs with Burst time and Priorities.

In section 4.2 the performance evaluation of between GA and MGA is discussed in which MGA performance is much better then GA in almost cases, same in section 4.3 the performance evaluation of between ABC and MABC is discussed in which MABC performance is much better then ABC in almost cases. In section 4.4 the performance evaluation of between all four GA, MGA, ABC and MABC is discussed in which MABC performance is much better than other all three (GA, MGA and ABC) in almost cases.

The future work is intended to an efficient hybrid approximation algorithm that will be treated as decision making system and support apply in recommendation system.

References

[1] D. E. Goldberg,Genetic Algorithms in Search. Optimization andMachine Learning, Addison Wesley, 1989.

[2] J. H. Holland, *Adaptation in Natural and Artificial Systems*. AnnArbor, MI: Univ. Michigan Press, 1975.

[3] K. A. De Jong, "Adaptive system design: A genetic approach," *IEEETrans. Syst., Man, Cybern.*, vol. SMC-10, pp. 566–574, 1980.

[4] J. E. Biegel and J. J. Davern, "Genetic algorithms and job-shop scheduling," *Comput. Ind. Eng.*, vol. 19, pp. 81–91, 1990.

[5] J. W. Herrmann and C. Y. Lee, "Solving a class scheduling problem with a genetic algorithm," Dept. Ind. Syst. Eng., Univ. Florida, Working Paper, Gainesville, 1993.

[6] J. W. Herrmann, C. Y. Lee, and J. Hinchmann, "Global job-shop scheduling with a genetic algorithm," Dept. Ind. Syst. Eng., Univ.Florida, Gainesville, 1994.

[7] J. C. Bean, "Genetics and random keys for sequencing and optimization," *ORSA J. Comput.*, vol. 6, pp. 154–160, 1994.

[8] Wen sheng Yao,et al. Genetic Scheduling on Minimal Processing Ele-mentsinthe Grid. Springer VerlagHeidelberg. 2002.

[9] Carole Fayad and Sanja Petrovic," A Genetic Algorithm for the Real-World Fuzzy Job Shop Scheduling ", Innovations in Applied Artificial Intelligence , Springer, Pages: 524–533, 2005.

[10] Vikas Gaba,Anshu Parashar. “Comparison of processor schedulingalgorithms using genetic approach”. International Journal of AdvancedResearch in Computer Science and Software Engineering 2 (8), August-2012, pp. 37-45.

[11] Lu Huang, Hai-shan Chen and Ting-ting Hu, "Survey onResource Allocation Policy and Job Scheduling Algorithms ofCloud Computing" ,Journal of Software, Vol. 8, No. 2, February2013, pp. 480-487.

[12] Fatos Xhafa, Ajith Abraham, “Computational models and heuristic methods for Grid scheduling problems”, “Future Generation Computer Systems 26”, 2010, pp.608-621.

[13] D. B. Fogel, “Evolutionary algorithms in theory and practice,” *Complexity*, vol. 2, no. 4, pp. 26–27, 1997.[Online]. Available: http://dx.doi.org/10.1002/(SICI)10990526(199703/04)2:4¡26::AID-CPLX6¿3.0.CO;2-7.

[14] H. Li, L. Wang, and J. Liu, “Task scheduling of computational grid based on particle swarm algorithm,” in *Computational Science and Optimization (CSO), 2010 Third International Joint Conference on*, vol. 2, may 2010, pp. 332–336.

[15] M. Houshmand, E. Soleymanpour, H. Salami, M. Amerian,and H. Deldari, “Efficient scheduling of task graphs to multiprocessors using a combination of modified simulated annealing and list-based scheduling,” in *Intelligent Information Technology and Security Informatics (IITSI), 2010 Third International Symposium on*, April 2010, pp. 350–354.

[16] K. Li, G. Xu, G. Zhao, Y. Dong, and D. Wang, “Cloud task scheduling based on load balancing ant colony optimization”, in *China grid Conference (China Grid), 2011 Sixth Annual*, aug. 2011, pp. 3–9.

[17] R. Shanmugapriya, S. Padmavathi, and S. Shalinie, “Contentionawareness in task scheduling using tabu search,” in *Advance Computing Conference, 2009. IACC 2009. IEEE International*, march 2009, pp. 272–277.

[18] Y. W. Wong, R. Goh, S.-H. Kuo, and M. Low, “A tabu search for the heterogeneous dag scheduling problem,” in *Parallel and Distributed Systems (ICPADS), 2009 15th International Conference on*, dec. 2009, pp. 663–670.

[19] http://www.geckil.com/~harvest/routing/image002.jpg.

[20] Kennedy, J. and Eberhart, R., “Particle Swarm Optimization,” Proceedings of the IEEE International Conference on Neural Networks, Perth, Australia 1995, pp. 1942-1945.

[21] Dorigo M. , Optimization, Learning and natural algorithms , Ph.D Thesis, Dip. Elect-tronica e Informazion, Politecnico di Milano Italy, 1992.

[22] D.T.Pham, A.Ghanbarzadeh, E.Koc, S.Otri, S.Rahim and M.Zaidi, the Bees Algorithm. Manufacturing Engineering Centre, Cardiff University, United Kingdom, 2005.

[23] D. Karaboga, An Idea Based On Honey Bee Swarm For Numerical Numerical Optimization, Technical Report-TR06 (Er ciyes University, Engineering Faculty, Computer Engineering Department), 2005.

[24] M. S. Kiran and M.G. Unduz, Int. J. Innov. Comput. Inf. Control 8, 6107 (2012).

[25] R. Chaukwale and S. S. Kamath, “A modified ant colony optimization algorithm with load balancing for job shop scheduling,” in Proceedings of the 15th International Conference on Advanced Computing Technologies (ICACT ’13), pp. 1–5, Rajampet, India, September 2013.

[26] Tawfeek, M.A., El-Sisi, A., Keshk, A.E., Torkey, F.A.: An Ant Algorithm for cloud task scheduling. In: Proceedings of International Workshop on Cloud Computing and Information Security (CCIS), pp. 169–172 (2013).

[27] DENG Guanlong, XU Zhenhao and GU Xingsheng, “A Discrete Artificial Bee Colony Algorithm for Minimizing the Total Flow Time in the Blocking Flow Shop Scheduling”, PROCESS SYSTEMS ENGINEERING *Chinese Journal of Chemical Engineering*, 20(6) 1067—1073 (2012).

[28] A. Madureira, I. Pereira, A. Abraham, " Towards Scheduling Optimization through Artificial Bee Colony Approach", IEEE 5th World Congress on Nature and Biologically Inspired Computing, USA, 2.

[29] M. S. Kiran and A. Babalik, J. Comput. Commun. 2 , 108 (2014).

[30] Ana Madureira, Bruno Cunha and Ivo Pereira, "Cooperation Mechanism for Distributed Resource Scheduling Through Artificial Bee Colony Based Self-Organized Scheduling System", 2014 IEEE Congress on Evolutionary Computation (CEC) July 6-11, 2014, Beijing, China.

[31] http://imgs.g4estatic.com/scheduling/SH1.jpg.