Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X

IJCSMC, Vol. 2, Issue. 7, July 2013, pg.333 – 343

RESEARCH ARTICLE

Genetic Algorithm Assisted Preventive Maintenance Scheduling for Web Servers

Ravi Sindal

Institute of Engineering and Technology, Devi Ahilya University, Indore, India
ravisindal@yahoo.com, ravisindal@gmail.com

Abstract— In this paper, A Preventive Maintenance Scheduling Problem of scheduling a set of Web servers located at centralized facility is addressed. The paper proposed a way of identifying an optimized solution for maintenance scheduling of the Web servers through Genetic algorithm. The performance parameter for denoting the capacity of Web servers is taken as requests/min. It is taken in to account that the net reserve of the installed servers must be greater than or equal to zero at any time slot of maintenance. The problem is formulated by developing chromosomes and fitness function. The Genetic algorithm is later tuned for varying number of chromosomes, generation and mutation rate to obtain the optimized maintenance schedule having minimum net reserve.

Key Terms: - Genetic Algorithm; Preventive maintenance; Web server; chromosomes; Artificial intelligence

I. INTRODUCTION

Preventive maintenance scheduling is among the most important problems faced by productive/service organizations. The preventive maintenance applied by servicing the equipment on regular intervals is for the purpose of increasing its reliability as much as possible. The problem has attracted researchers due to its economic importance and complexity. Preventive Maintenance of Web servers located at centralized facility is also emerging as one of the application in this category. The Web servers running for 24 hours a day, 365 days a year need regular shutdown and maintenance to extend their life and improving their performance. Several techniques have been used to layout the preventive maintenance plan for the Web servers. Genetic algorithm (GA) has recently been introduced to find the optimized solution for their maintenance scheduling [3] [4].

The paper illustrates a virtual Centralized Web server facility comprising of seven Web servers, Each Web server has individual capacity of handling requests generated per minute. The idea is to divide the preventive maintenance scheduling in four time slots so that the normal working go on uninterrupted while maintaining a minimum net reserve of requests/min. Section II of the paper refers to the benchmarking parameters for a Web server. We have included requests/min as the performance parameter in our research. Section III of the paper illustrates about the theory of Genetic algorithm. Section IV gives a view of GA process followed to obtain optimized maintenance scheduling. Section V and VI discusses about the result and conclusion obtained.

II. WEB SERVER BENCHMARKING

The term web server refers to either the hardware (the computer) or the software (the computer application) that helps to deliver web content that can be accessed through the Internet. The primary function of a web server is to cater web page to the request of clients using the Hypertext Transfer Protocol (HTTP). This means delivery of HTML documents and any additional content that may be included by a document, such as images, style

sheets and scripts. Web server benchmarking is the process of estimating a web server performance in order to find if the server can serve sufficiently high workload. The performance is usually measured in terms of [6]:

Requests per Second (RPS)

RPS is the evaluation of how many requests per second are being sent to a target server. In other words, this metric is called Average Load and it allows you to understand what load your web application currently works under. Usually, this is calculated as a count of the requests received during a measurement period, where the period is represented in seconds.

Error Rates

The Error Rate is usually calculated as a percentage of problem requests relative to all requests, and it reflects how many response HTTP status codes indicate an error on the server – including any requests that never get a response (timed out). Error Rate is a significant metric because it measures "performance failure" in the application and tells you how many failed requests have occurred at a particular point in time.

Average Response Times (ART)

By measuring the duration of every request/response cycle, one can be able to evaluate how long it takes the target web application to generate a response. The ART takes into consideration every round trip request/response cycle during a Monitoring period and calculates the mathematical mean of all the response times.

Peak Response Times (PRT)

PRT also measures the round trip of request/response cycles, however the peak will tell us what the longest cycle is at that point in the test. The standard measurement unit of PRT is recommended to be milliseconds.

Inbound and Outbound throughput

Throughput is the measurement of bandwidth consumed during a request. It shows how much data is flowing back and forth from your servers. Usually the throughput is measured in units of Kilobytes per Second.

Uptime

Uptime is the amount of time that a server has stayed up and running properly. It reflects the reliability and availability of the server and, obviously, this value should be as large as possible. The value can be calculated as an absolute value or as a percentage of actual server uptime to ideal server uptime.

CPU utilization

CPU Utilization is the amount of CPU time used by the Web Application while processing a request. Usually, it is the percentage of CPU usage that is calculated, which indicates how much of the processor's capacity is currently in use by your application.

Memory utilization

Memory Utilization refers to the amount of memory used by a Web Application while processing a request. Usually, it is calculated as the process's percentage of memory utilization.

The Count of threads

As usual, a web application can generate a lot of threads to process requests. The number of threads is an important metric because the number of threads per process is normally limited by the system. So if an application generates too many threads it can be an indicator that there is a problem in the application. Obviously, the count of existing threads is proportional to the load and inversely proportional to the processing time of the requests.

III. GENETIC ALGORITHM AND GA'S IN SCHEDULING

Genetic algorithms are a class of stochastic search algorithms based on biological evolution. Given a clearly defined problem to be solved and a binary string representation for candidate solutions, a basic GA can be represented as in Fig.1. A GA applies the following major 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 p_c and the mutation probability p_m.

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

Step 3: Randomly generate an initial population of chromosomes of size N:

$$x_1, x_2, \ldots, x_N$$

Step 4: Calculate the fitness of each individual chromosome:

$$f(x_1), f(x_2)\ldots\ldots\ldots f(x_N)$$

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 [1][2].

Step 6: Create a pair of offspring chromosomes by applying the genetic operators – crossover and mutation.

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

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

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

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

As we see, a GA represents an iterative process. Each iteration is called a generation. A typical number of generations for a simple GA can range from 50 to over 500. The entire set of generations is called a run. At the end of a run, we expect to find one or more highly fit chromosomes.

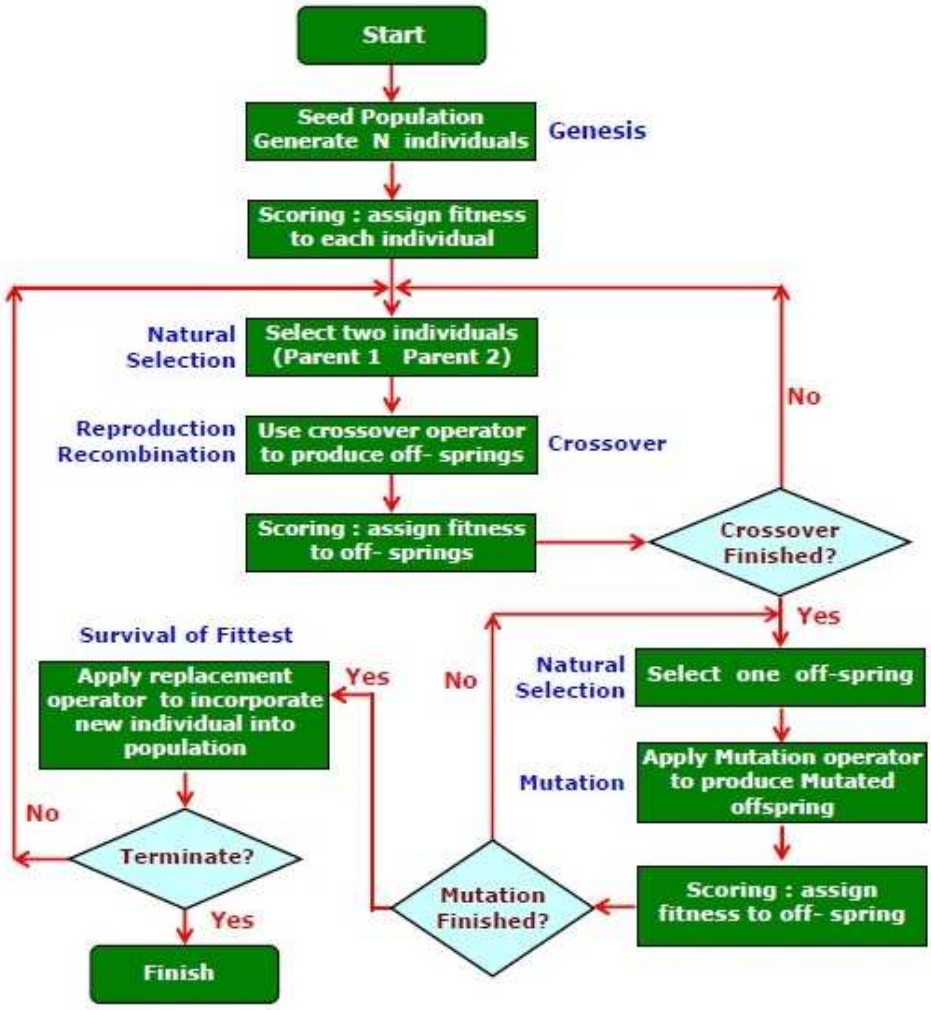

Fig.1 Genetic Algorithm Flowchart

IV. MAINTENANCE SCHEDULE MODEL FOR WEB SERVERS

Preventive maintenance means the care and servicing by personnel for the purpose of maintaining equipment and facilities in satisfactory operating condition by providing for systematic inspection, detection, and correction of incipient failures either before they occur or before they develop into major defects. The primary goal of

maintenance is to avoid or mitigate the consequences of failure of equipment. This may be by preventing the failure before it actually occurs which Planned Maintenance and Condition Based Maintenance help to achieve. It is designed to preserve and restore equipment reliability by replacing worn components before they actually fail. The ideal preventive maintenance program would prevent all equipment failure before it occurs. Preventive maintenance of Web servers includes the timely shut down of running servers. During the shutdown process various hardware and software testing are done to see its working. The faults and errors found during testing are corrected later [1][4].

The problem we discuss here is the maintenance scheduling in Web servers located at central facility. This task has to be carried out under several constraints and uncertainties, such as failures and forced outages of running Web servers and delays in obtaining spare parts. The schedule often has to be revised at short notice. Human experts usually work out the maintenance scheduling by using different software and hardware routines, and there is no guarantee that the optimum or even near-optimum schedule is produced.

The steps in maintenance scheduling of Web servers using GA is discussed in following steps:

Step 1: Problem Specification, Constraints and optimum criteria

This is probably the most important step in developing a GA, because if it is not correct and complete a viable schedule cannot be obtained. Web Servers are made to operate continuously throughout their life by means of preventive maintenance. The purpose of maintenance scheduling is to find the sequence of outages of Web servers over a given period of time (normally a year) such that the availability of service is maximized. Any outage in Web servers is associated with some loss in availability of service. The availability margin is determined by the system's net reserve. The net reserve, in turn, is defined as the total installed server capacity of the system minus the server capacity lost due to a scheduled outage and minus the maximum load forecast during the maintenance period. For instance, if we assume that the total installed capacity is $150X10^5$ requests /min and a unit of $20X10^5$ requests /min is scheduled for maintenance during the period when the maximum load is predicted to be $100X10^5$ requests /min, then the net reserve will be $130X10^5$ requests /min. Maintenance scheduling must ensure that sufficient net reserve is provided for Web services during any maintenance period.

We are considering a case where seven Web Servers to be maintained in four equal intervals. The maximum loads expected during these intervals are $60X10^5$, $80X10^5$, $60X10^5$ and $55X10^5$ requests /min. The Web server capacities and their maintenance requirements are presented in Table 1.

TABLE I
Web Servers and Their Maintenance Requirements

Server No.	Request served per minute (X 10^5)	No. of Time slots required for server maintenance/Year
1	25	1
2	15	1
3	30	2
4	40	2
5	10	1
6	15	1
7	15	1

Step 2: Converting Problem domain in Chromosome form

We have assumed that maintenance of any server starts at the beginning of an interval and finishes at the end of the same or adjacent interval. The maintenance cannot be aborted or finished earlier than scheduled. Also the net reserve of the installed servers must be greater than or equal to zero at any interval. The optimum criterion here is that the net reserve must be at the maximum during any maintenance period. The unit schedule can be easily represented as a 4-bit string, where each bit is a maintenance interval. If a unit is to be maintained in a particular interval, the corresponding bit assumes value 1, otherwise it is 0. It also shows that the number of intervals required for maintenance of this unit is equal to 1. The sequence of maintenance schedules of individual servers can be represented as Fig.2.

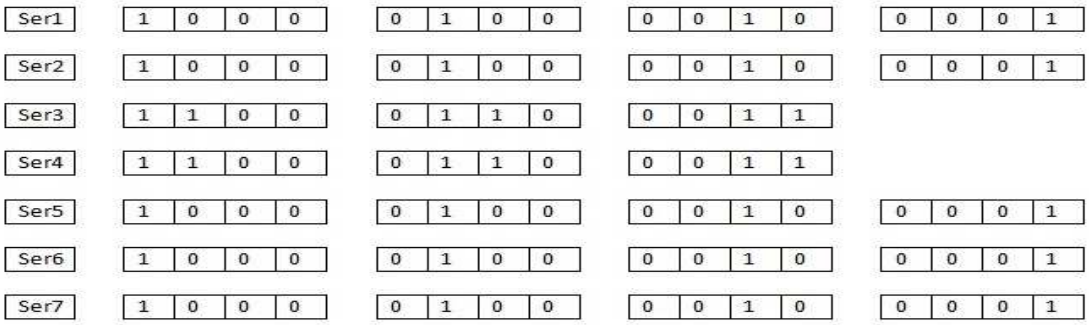

Fig no.2 Sequence of maintenance schedules of individual servers

Also, a complete maintenance schedule for our problem can be represented as a 28-bit chromosome. A sample of such a chromosome is shown in Fig.3.

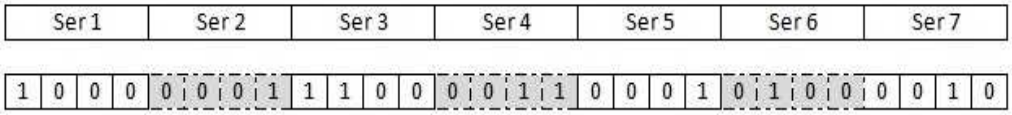

Fig no.3 A chromosome for Maintenance scheduling

Step 3: Defining fitness function to evaluate the chromosome performance

The fitness function must capture what makes a maintenance schedule either good or bad for the user. The evaluation of a chromosome starts with the sum of capacities of the Web servers scheduled for maintenance at each interval. For the chromosome shown in Fig.3, we obtain [1][3]:

Interval 1: 1 X ($25X10^5$) + 0 X ($15X10^5$) + 1 X ($30X10^5$) + 0 X ($40X10^5$) +0 X ($10X10^5$) + 0 X ($15X10^5$) + 0 X ($15X10^5$) =$55X10^5$

Interval 2: 0 X ($25X10^5$) + 0 X ($15X10^5$) + 1 X ($30X10^5$) + 0 X ($40X10^5$) +0 X ($10X10^5$) + 1 X ($15X10^5$) + 0 X ($15X10^5$) =$45X10^5$

Interval 3: 0 X ($25X10^5$) + 0 X ($15X10^5$) + 0 X ($30X10^5$) + 1 X ($40X10^5$) +0 X ($10X10^5$) + 0 X ($15X10^5$) + 1 X ($15X10^5$) =$55X10^5$

Interval 4: 0 X ($25X10^5$) + 1 X ($15X10^5$) + 0 X ($30X10^5$) + 1 X ($40X10^5$) +1 X ($10X10^5$) + 0 X ($15X10^5$) + 0 X ($15X10^5$) =$65X10^5$

Then these values are subtracted from the total installed capacity of Web servers (in our case, 150 X 10^5 requests/min):

Interval 1: (150-55) X 10^5= 95 X 10^5

Interval 2: (150-45) X 10^5= 105 X 10^5

Interval 3: (150-55) X 10^5= 95 X 10^5

Interval 4: (150-65) X 10^5= 85 X 10^5

And finally, by subtracting the maximum loads expected at each interval, we obtain the respective net reserves:

Interval 1: (95-60) X 10^5= 35 X 10^5

Interval 2: (105-80) X 10^5= 25 X 10^5

Interval 3: (95-60) X 10^5= 35 X 10^5

Interval 4: (85-55) X 10^5= 30 X 10^5

Since all the results are positive, this particular chromosome does not violate any constraint, and thus represents a legal schedule. The chromosome's fitness is determined as the lowest of the net reserves; 25 X 10^5 requests/min

Step 4: Constructing genetic operators.

Each gene in a chromosome is represented by a 4-bit indivisible string, which consists of a possible maintenance schedule for a particular Web server. Thus, any random mutation of a gene or recombination of several genes from two parent chromosomes may result only in changes of the maintenance schedules for individual units, but cannot create 'unnatural' chromosomes.

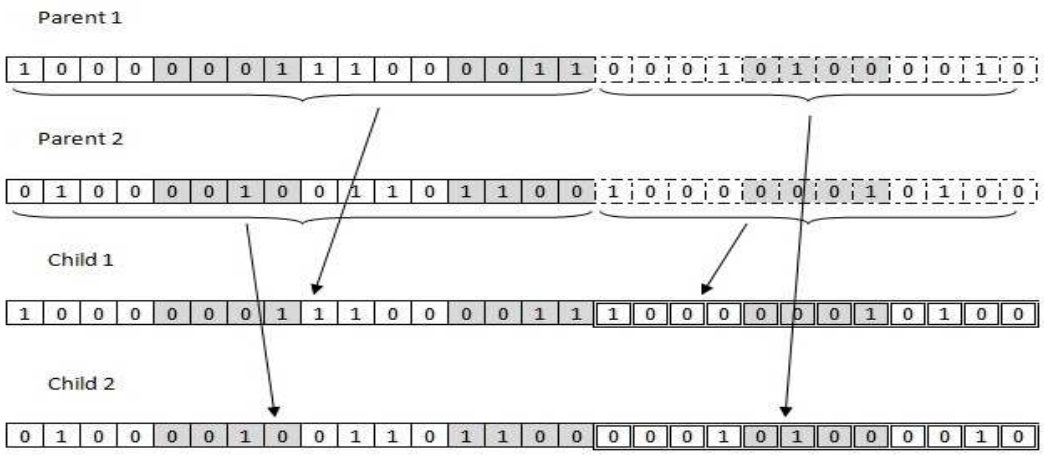

Fig no.4 Crossover Operation for Maintenance Scheduling

Fig.4 shows an example of the crossover application during a run of the GA. The children are made by cutting the parents at the randomly selected point denoted by the vertical line and exchanging parental genes after the cut.

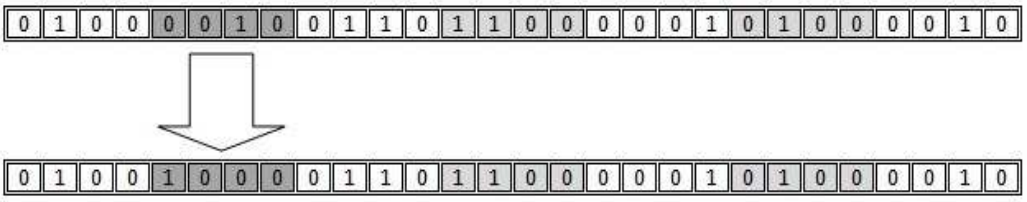

Fig no.5 Mutation Operation for Maintenance Scheduling

The mutation operator randomly selects a 4-bit gene in a chromosome and replaces it by a gene randomly selected from the corresponding pool. This is shown in Fig.5.

Step 5: Running GA and tuning

For running the GA, we must choose the population size and the number of generations to be run. Larger population can achieve better solutions than a smaller one, but will work more slowly. In fact, however, the most effective population size depends on the problem being solved, particularly on the problem coding scheme. The GA can run only a finite number of generations to obtain a solution [1] [5].

V. RESULTS AND DISCUSSIONS

Fig.6 and 7 presents performance graphs and the best schedule created by 50 generations of 20 chromosomes. As noticed, the minimum of the net reserves for the best schedule is zero, which is not acceptable. We now increase the number of generations to 100 and compare the best schedules. Fig.8 and 9 shows the results. The best schedule now provides the minimum net reserve of 30 X 10^5 request/min. However, in both cases, the best individuals appeared in the initial generation, and the increasing number of generations did not affect the final solution. It indicates that we should try increasing the population size. Fig.10 and 11 shows fitness function values across 100 generations, and the best schedule so far. There is no change in the minimum net reserve. To the get the optimized solution let us increase the mutation rate to 0.01 and rerun the GA once more. Fig.12, 13 and 14 represents the results. The minimum net reserve is now 25 X 10^5requests/min. We got two identical solutions- one showing maintenance schedule of Web server 3 in time slot 1 & 2 and maintenance schedule of Web server 4 in time slot 3 and 4 as shown in Fig.13. The other one showing maintenance schedule of Web server 3 in time slot 3 & 4 and maintenance schedule of Web server 4 in time slot 1 and 2 as shown in Fig.14 . We can use either of the two optimized maintenance schedules as required.

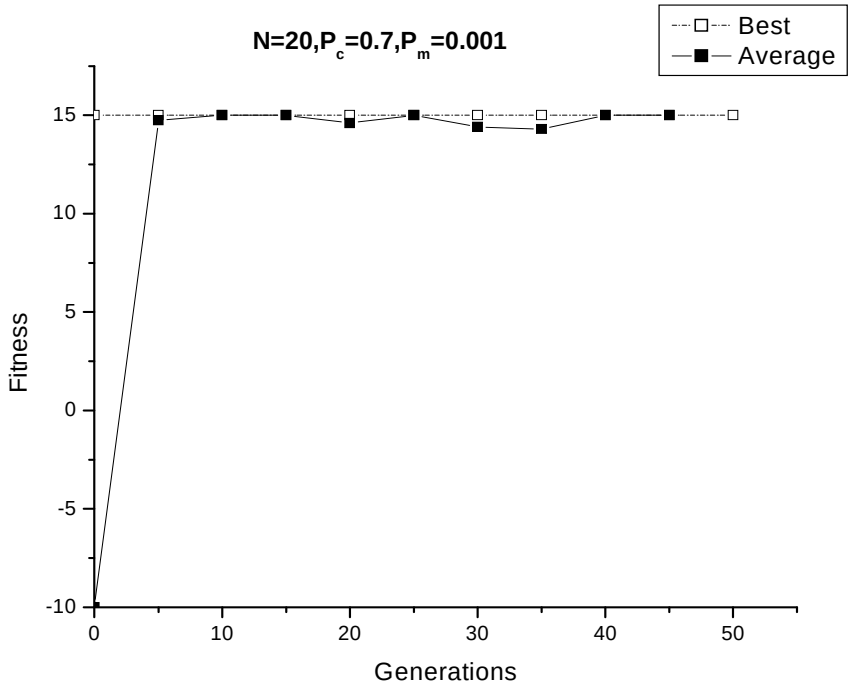

Fig no.6 Performance Graph for Population of 20 Chromosomes and 50 Generations

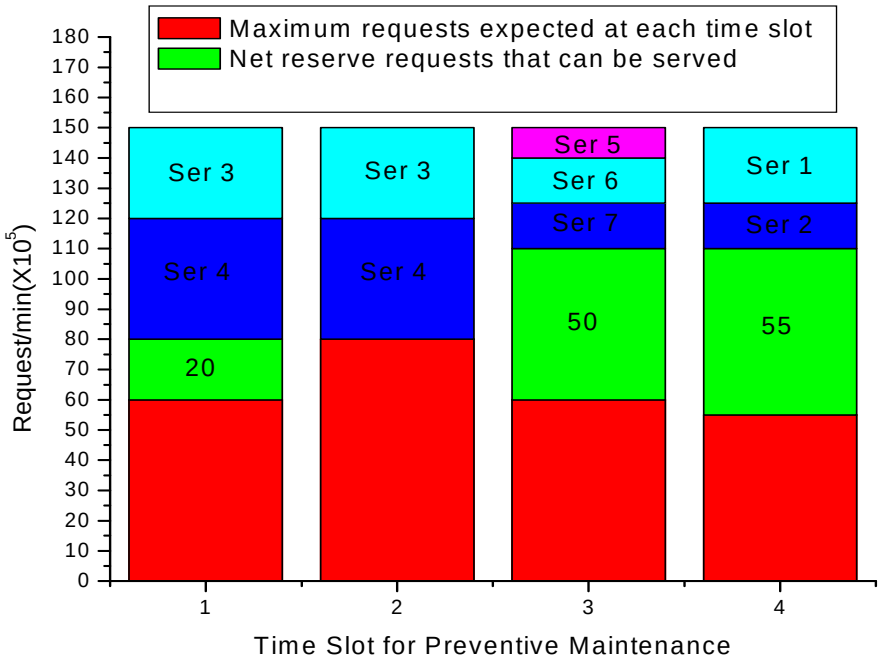

Fig no.7 Best Maintenance Schedule for Population of 20 Chromosomes and 50 Generations

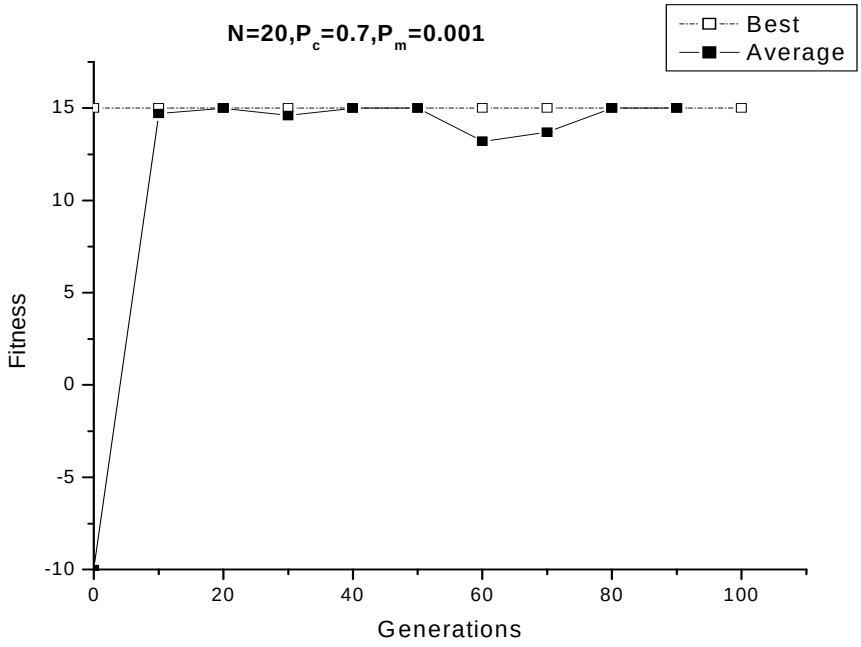

Fig no.8 Performance Graph for Population of 20 Chromosomes and 100 Generations

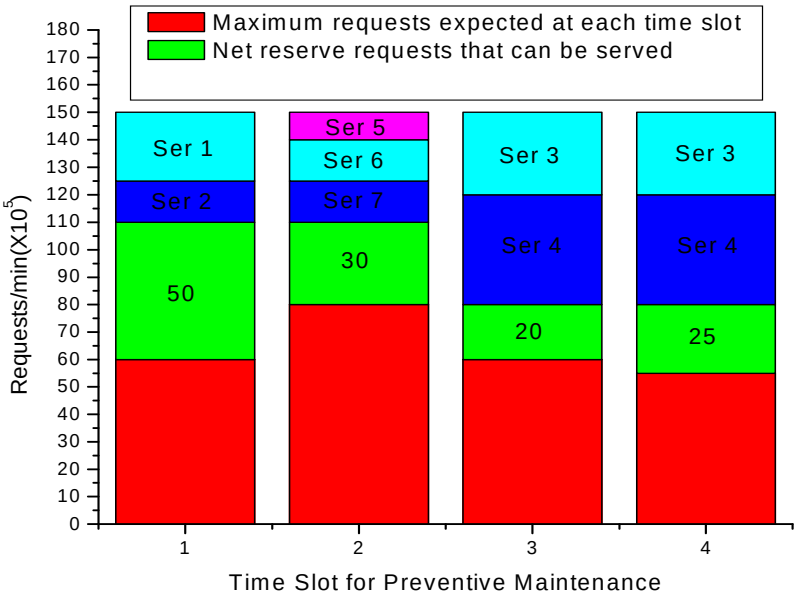

Fig no.9 Best Maintenance Schedule for Population of 20 Chromosomes and 100 Generations

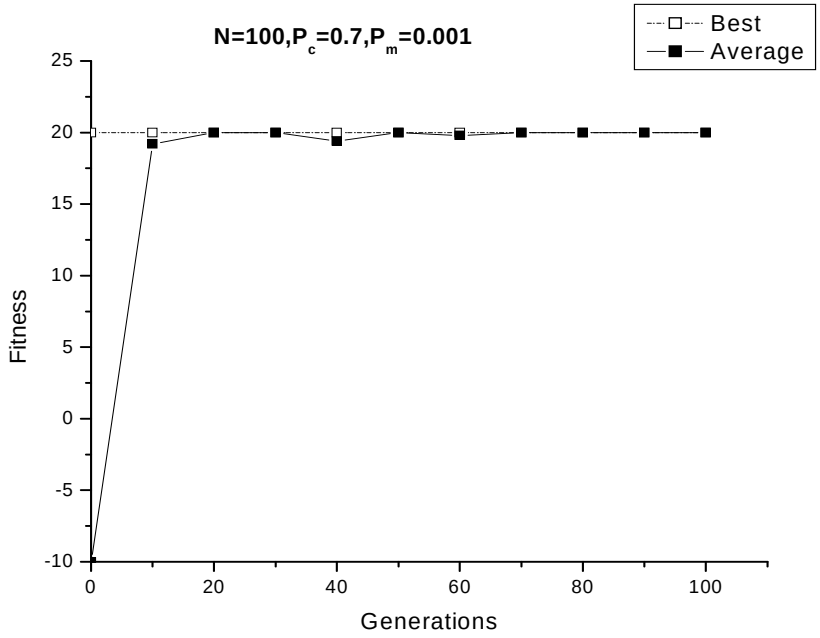

Fig no.10 Performance Graph for Population of 100 Chromosomes, 100 Generations, Mutation rate=0.001

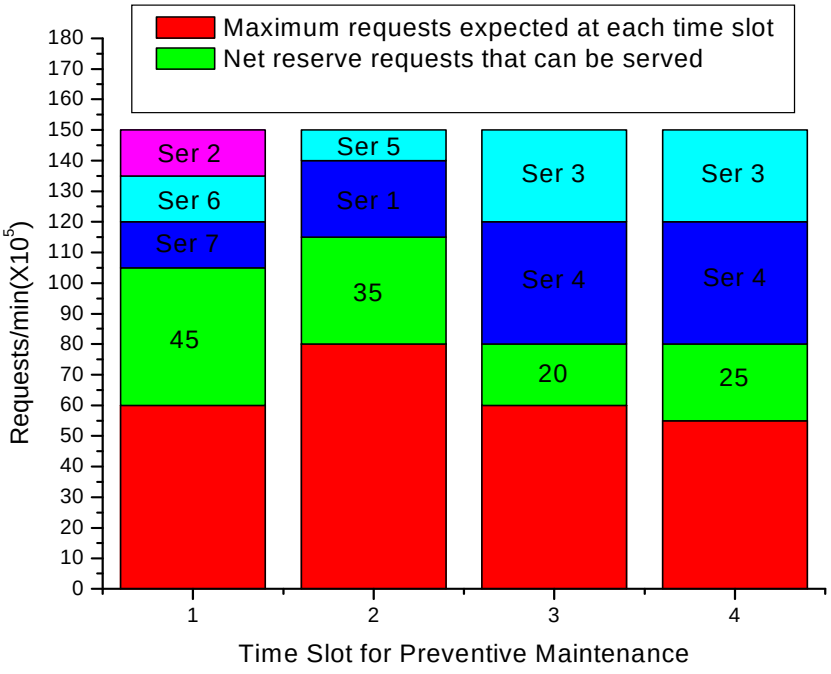

Fig no.11 Best Maintenance Schedule for Population of 100 Chromosomes, 100 Generations, Mutation rate=0.001

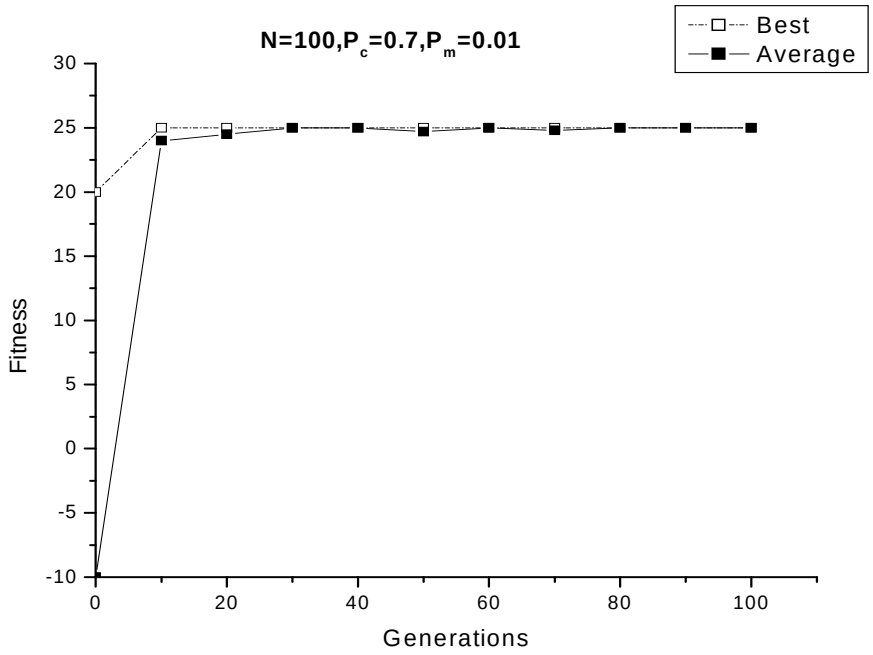

Fig no.12 Performance Graph for Population of 100 Chromosomes, 100 Generations, Mutation rate=0.01

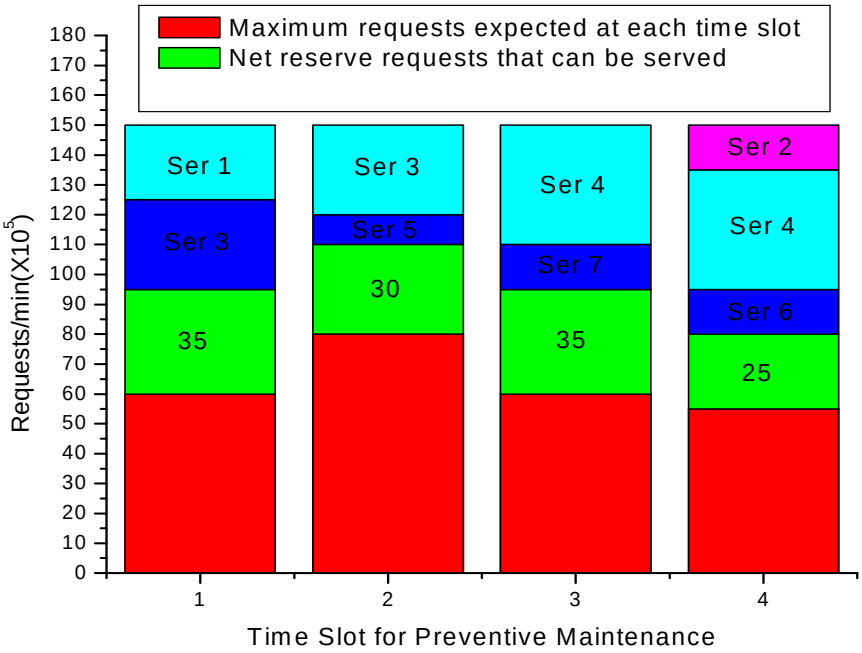

Fig no.13 Best Maintenance Schedule for Population of 100 Chromosomes, 100 Generations, Mutation rate=0.01

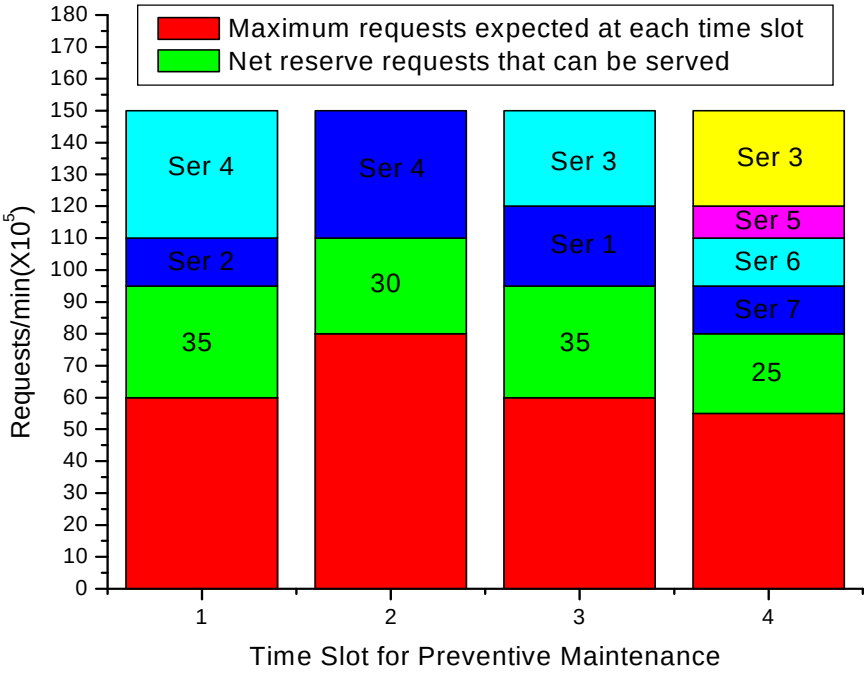

Fig no.14 Best Maintenance Schedule for Population of 100 Chromosomes, 100 Generations, Mutation rate=0.01

VI. CONCLUSIONS

The conclusion that can be drawn from GA assisted maintenance scheduling of Web servers are summarized as:

- GA offers a better mechanism of obtaining optimized scheduling.
- Increasing the number of chromosomes and generation lead to near optimized results.
- The mutation rate can be varied to see the effect on obtained results.

REFERENCES

[1] Micheal Negnevitsky, "Artificial Intelligence- A Guide toIntelligent system," Addison Wesley,2005.
[2] Mohamed Ali Abdel-Fattah Mansour, "Solving the Periodic Maintenance scheduling Problem via Genetic Algorithm to Balance Workforce Levels and Maintenance Cost", American J. of Engineering and Applied Sciences 4 (2): pp 223-234, 2011
[3] K.P Dahal,C.J Aldbridge,,J.R McDonald," Generator Maintenance scheduling using Genetic Algorithm using Fuzzy evaluation function" Fuzzy sets and systems, Elsevier Publication,102,pp 21-29,1999.
[4] Liang Sun, Xiaochun Cheng, Yanchun Liang, "Solving Job Shop Scheduling Problem Using Genetic Algorithm with Penalty Function", International Journal of Intelligent Information Processing Volume 1, Number 2, pp 65-77,December 2010.
[5] Seungchul Lee and Jun Ni, "Genetic Algorithm for Job Scheduling with Maintenance Consideration in Semiconductor Manufacturing Process," Mathematical Problems in Engineering ,Hindawi Publishing Corporation ,2012
[6] www.wikipedia.com

