A Case Study of Parameter Control in a Genetic Algorithm: Computer Network Performance

J.A. Fernández-Prieto[1], J. Canada-Bago[1], M.A. Gadeo-Martos[1], and Juan R. Velasco[2]

[1] Telecommunication Engineering Department, E.P.S. Linares, University of Jaén, Alfonso X El Sabio, 28, 23700 Linares (Jaén)
{jan,jcbago,gadeo}@ujaen.es
[2] Departament of Automatic, University of Alcalá, Campus Universitario, 28871 Alcala de Henares (Madrid)
juanra@aut.uah.es

Abstract. Genetic Algorithms use different parameters to control their evolutionary search for the solution to problems. However, there are no standard rules for choosing the best parameter values, being difficult to know whether the parameter values must be fixed during a run or must be modified dynamically. Besides, there are many theoretical results on parameter control, but however, very often real world problems call for shortcuts and/or some *ad hoc* solutions. This paper presents an effective approach for optimization of control parameters which is based on a meta-GA combined with an adaptation strategy to improve the GA performance. In order to validate the approach, it has been applied to verify the performance of a real system: a computer network. The results have been compared with the ones obtained for other methods: using fixed and adapted parameter values. A statistical analysis has been done to ascertain whether differences are significant.

Keywords: Parameter control, Computer Networks, Throughput.

1 Introduction

Genetic Algorithms (GAs) are search algorithms based on natural genetics that provide robust search capabilities in complex spaces, and thereby offer a valid approach to problems requiring efficient and effective search processes [1]. The basic idea is to maintain a population of individuals (representing candidate solutions to the problem) that evolves over time through a process of competition.

There are two basic models to implement a GA. The *Generational (or canonical) GA* [1] uses non-overlapping populations so the entire population is replaced by the new generation offspring whilst a S*teady-State GA* [2] operates on overlapping populations in which only a subset of the current population is replaced in each generation. In both cases, the behaviour of the GAs is strongly determined by the balance between exploration (to investigate new and unknown areas in a search space) and exploitation (to make use of knowledge acquired by exploration to reach better positions on the search space) [3]. GAs use a number of parameters to control their evolutionary search for the

P. Chatzimisios et al. (Eds.): MobiLight 2010, LNICST 45, pp. 650–661, 2010.

solution to their given problems. The GA control parameter settings, such as population size, N, probability of crossover, Pc, probability of mutation, Pm, probability of selection, Ps, (to select the individuals from the population in order to use them as parents for the new offspring), are key factors in the determination of the exploitation versus exploration tradeoff. These parameters greatly determine whether the algorithm will find a near-optimum solution, and whether it will find such a solution efficiently. It has long been acknowledged that they have a significant impact on GA performance [4]. However, there are no hard and fast rules for choosing appropriate values for these parameters and many researchers based their choices on tuning the control parameters "by hand", that is, experimenting with different values and selecting the ones that gave the best results (as a problem of trial and error). Later, they reported their results of applying a GA to a particular problem, paraphrasing: ". . . for this experiment, we have used the following parameters: population size of 50, probability of crossover equal to 0.95, etc." without much justification of the choice made [5].

The problem of finding optimal control parameters for GAs has been studied by many authors, from the first until the last recent studies: [3][6][7][8][9][10][11], etc. Michalewicz and Schmidt [11] indicate that there are many theoretical and experimental results on parameter tuning and parameter control, but however, very often real world problems call for shortcuts and/or some *ad hoc* solutions.

According to the *No Free Lunch* theorem [12], an algorithm has different performance on different problems, so an optimal or near-optimal set of control parameters for one GA does not generalize to all cases. This stresses the need for efficient techniques that help finding good parameter settings for a given problem.

Furthermore, different control parameter values may be necessary during the course of a run to induce an optimal exploration/exploitation balance. In [9] the authors argument that any static set of parameters, having the values fixed during a GA run, seems to be inappropriate, and it is desirable to dynamically change parameter values. For instance, large mutation steps can be good in the early generations helping the exploration of the search space while small mutation steps might be needed in the late generations to help fine tuning the optimal individuals. Note that a GA is an intrinsically dynamic, adaptive process. Therefore it is a natural idea to try to modify the control parameter values during the run of the GA. It is possible to do this by using some (possibly heuristic) rules, by taking feedback from the current state of the search, or by employing some self-adaptive mechanism. These changes may affect to a single component of an individual, the whole individual, or even the whole population. Clearly, by changing these values while the algorithm is searching for the solution of the problem, further efficiencies can be gained [13]. For these reasons, Adaptive Genetic Algorithms (AGAs) have been built. They dynamically adjust selected control parameters or genetic operators during the course of evolving a problem solution, offering the most appropriate exploration and exploitation behaviour. The straightforward way to treat this problem is by using parameters that may change over time, that is, by replacing a parameter p by a function p(t), where t is the generation counter. However, if the problem of finding optimal static parameters can be quite difficult, designing an optimal function p(t) may be even more difficult.

However, De Jong [14] indicates that it is a difficult thing to ascertain whether there is any advantage to be gained by dynamically changing the value of a parameter during an GA run and, if so, how to change it. What can we do? Should parameter

values be fixed during a run or be modified dynamically? How do changes in a parameter affect the performance of a GA? How does one choose appropriate parameter values?

In this paper we present an approach based on a meta-GA, which is combined with an adaptation strategy of the GA control parameter to find and adjust the optimum parameter values, to improve the GA performance. In order to validate the approach, it has been applied to verify the performance of a real system: a computer network. Different comparisons are performed, aiming to assess the acceptable optimization power of the proposed system. The results have been compared with the ones obtained for other methods: using fixed and adapted parameter values. Moreover, a statistical analysis has been done to ascertain whether differences are significant among the proposed system and the other algorithms.

The remainder of the paper is organized as follows. Section 2 describes a case study, the computer network performance, to illustrate the impact on the GA performance from the control parameter settings and presents the topology of the network exploited in the experiments. Section 3 shows the components of the GA. Section 4 provides the system used to optimize the GA control parameter. Results are reported in Section 5 and finally some conclusions are drawn in Section 6.

2 A Case Study: Computer Network Performance

Nowadays, it is important to test computer networks under realistic traffic loads. Poisson's statistical processes [15] are very often adopted as models of background traffic in order to model the computer network performance. In [16][17], the authors argue that Internet traffic can be well characterized by using Poisson models.

However, other authors [18][19] indicate that the use of Poisson models is a clever choice of background traffic pattern to yield useful results, and the worst/best case analysis cannot be done. Therefore it is necessary a mixed simulation technique based on a GA to explore the solution space. They propose to integrate a GA with a network simulator to drive the generation of critical background traffic. The GA aims at generating the worst-case traffic for the computer network under analysis and finds the traffic configuration that minimizes its performance, given some constraints on the traffic bandwidth. Baldi et al. [18] use the static control parameter values which are shown in Table 1 and Karthik et. al [19] use the shown in Table 2.

However, are these static control parameter values the best to optimize the performance of the GA? Which set of parameter values is better? Why not use an AGA? In this case, how the parameter values must be changed?

Table 1. GA parameter values used in [18]

Parameter	Value
Population size	50
Selection probability	0.4
Crossover probability	1.0
Mutation probability	0.01
Number of generations	500

Table 2. GA parameter values used in [19]

Parameter	Value
Population size	50
Selection probability	0.4
Crossover probability	1.0
Mutation probability	0.005
Number of generations	500

Figure 1 shows the architecture of the environment which integrates the GA and the simulator.

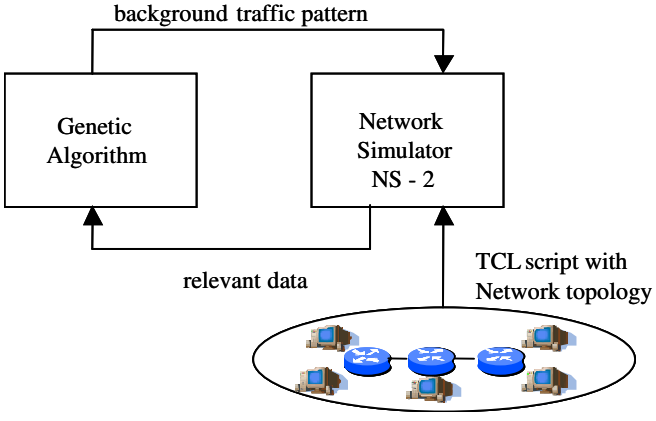

Fig. 1. Architecture of the environment that integrates the GA and the simulator

The approach is quite general and can be applied to different protocols using different simulators with limited effort. We have used a publicly available simulator called ns-2 [20]. It is a discrete event simulator targeted at networking research and provides substantial support for simulation of TCP, routing, and multicast protocols over wired and wireless (local and satellite) networks. The protocol to examine is TCP protocol in order to better focus on the verification methodology and exploitation of the GA and not on the protocol itself. The aim is to study the TCP protocol in real operating conditions. It set up an IP network and a number of TCP probe connections (sender-receiver pairs). The network is loaded with a background traffic generated by UDP (User Datagram Protocol) sender - receiver pairs. During the analysis process, the GA provides a pattern of the traffic generated by background sources and a network simulation is run on the given topology. During this simulation, relevant data are gathered from probe connections by the simulator program and provided to the GA, which uses them to estimate the damage or the benefit that the background traffic made. Such information is then used to drive the generation of traffic patterns to be used in subsequent steps.

2.1 Network Topology

The topology of the network exploited in the experiments is shown in Figure 2. Three TCP connections span from the transmitters TXi to the receivers RXi through three IP routers. Each TCP connection performs long file transfers generating a series of 1024 Kbyte messages at a maximum mean rate of 1.33Mb/s. Acknowledgments from each transmitter are carried by messages flowing in the reverse direction. These TCP connections represent the probe connections of the experiments.

Two sources (BSi) generate background UDP traffic directed to their respective destinations (BDi) over the same links traversed by the TCP connections. The timing

of background packets is controlled by the GA. Each link has a capacity of 10Mbps in each direction and introduces a fixed 10µs delay. Routers introduce a fixed 0,1 ms delay component which accounts for the processing required on each packet and adds to the queuing delay.

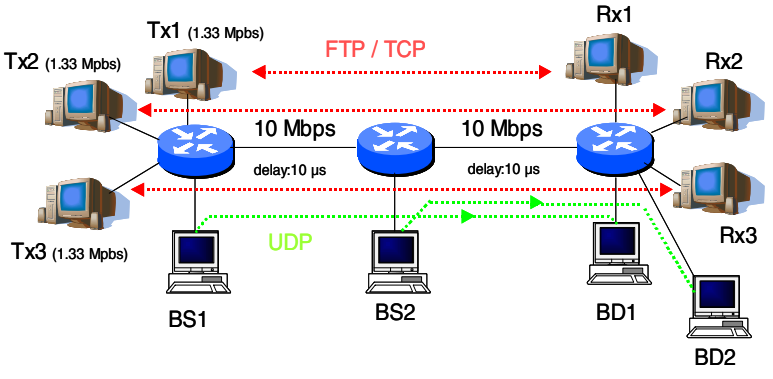

Fig. 2. Topology of the network

3 Genetic Algorithm

For a description of the GA used we have to bear in mind the following five components:

- Solution encoding. Each individual represents a background traffic pattern. Therefore each individual encodes the description of the traffic generated by all the background connections for the whole duration of the simulation. A connection is specified by a UDP source and destination pair. Individuals are encoded as strings of 4500 genes. Each gene represents a single background packet. Genes are composed of the *delay* that represents how long the given source will wait before sending a new packet after sending the current one.
- Initialization. The background traffic corresponding to the initial population is generated according to a Poisson process whose inter-arrival time between packets is exponentially distributed.
- Fitness evaluation. The fitness function measures the probe connections' throughput, i.e., the performance of the probe TCP connections perceived by end users during the simulation experiment. All bytes successfully received at the TCP level, but not delivered to end-users, such as duplicated packets received due to the retransmission mechanism of the protocol, are not considered. The fitness function should increase with the increasing goodness of a solution. As the throughput is being minimized, a solution is good when the background traffic pattern is critical. Therefore, the fitness function is inversely proportional to the total number of bytes perceived by the end users.
- Genetic Operators. The selection operator picks the individuals from the population in order to use them as parents for the new offspring. A selection of individuals of

the population is carried out based on the probability of selection. These individuals are selected according to the tournament selection. The GA applies one point crossover and a random mutation that substitutes each gene with a new random one. After each mutation, genes in the individual need to be sorted. The elitist strategy [6] is considered as well.

- GA parameter values. The parameters used by [18][19] for the GA are summarized in Tables 1 and 2. Instead, we propose to use other probabilities (that may change over time) which have been previously found using our system [21].

Some of the software for the GA used the GAlib genetic algorithm package, which has been written by Matthew Wall at the Massachusetts Institute of Technology [22]. GAlib contains a set of C++ genetic algorithm objects. The library includes tools for using genetic algorithms to do optimization in any C++ program.

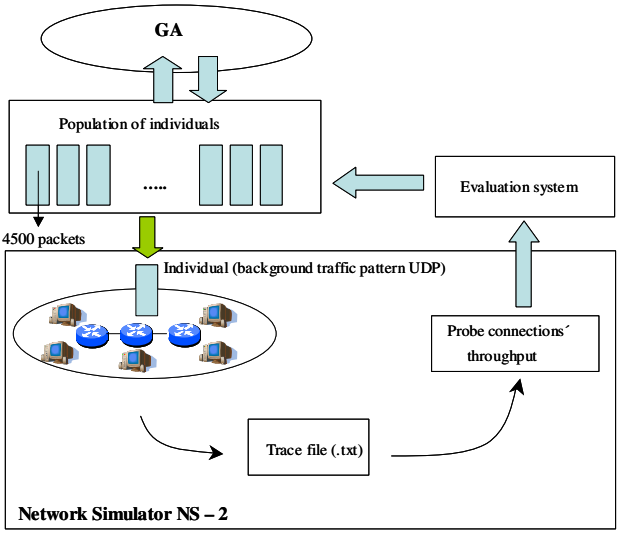

Fig. 3. Genetic Algorithm

4 Optimization of Control Parameters

In order to optimize the control parameters, we propose an approach which is based on a meta-GA (GA within a GA) combined with an adaptation strategy of the GA parameters. The population of the meta-GA consists of a group of GAs, in which each one is running under the same configuration of genetic operators (indicated in Section 3). Each GA can be a Generational GA or Steady-State GA, depending on the selection probability used. The meta-GA is applied to investigate evolving the parameter settings of genetic operators for the GAs.

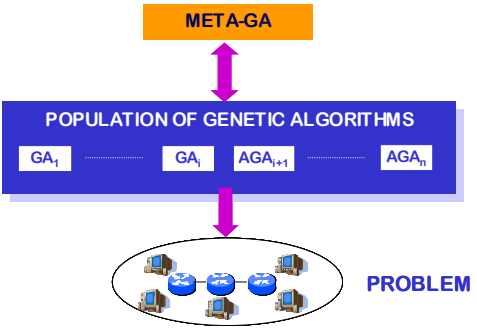

Fig. 4. Approach proposed

However, as indicated in Section 1, the aim of this work is to find the optimum probabilities to improve the GA performance, so really, each individual of the meta-GA population represents a set of probabilities applied to the genetic operators used by each GA. Therefore, we optimize the selection, crossover and mutation probabilities. We do not include the population size in order to establish a reference point to validate the approach. In this way, we can know the GA performance using the parameter values which have been found previously by the meta-AG, and using the parameter values that have been considered frequently in the GA literature [4][6].

The probabilities are dynamically adjusted and can change over time. We use three functions $p_i(t)$, i = 1, 2, 3 where each one affects to Ps, Pc and Pm independently. Each function has the following form:

$$p(t) = p_o^i\left(1 - \frac{(Ln(t+1))^{(1/a_i)}}{(c_i Ln(T+1))^{(1/a_i)}}\right) \qquad \begin{matrix} a_i \in]0,2] \\ c_i \in [1,T] \end{matrix} \tag{1}$$

where p_o^i is the initial value of the probability, t is the generation counter which shall be kept in the interval [0, T] (T is the maximum of generations). The parameter a_i models the curvature concavity or convexity whilst c_i drives the initial attenuation value. Figure 5 shows different curvatures depending on the values a_i and c_i for T=500 and p_o^i = 0.75. In order to optimize the probabilities of the GA, as well as the parameters a_i and c_i of each function $p_i(t)$, we apply a meta-GA, called Additional Genetic System (AGS) to the GA, as shown in Figure 7. In the population of the AGS a candidate solution is PS_n, n=1,…,20. Each one represents the probabilities set applied to the genetic operators which are used by the GA along with the parameters a_i and c_i. Therefore each PS_n codes a vector of real values in the following way:

$$PS_n = \{\ Ps, Pc, Pm, a1, c1, a2, c2, a3, c3\}$$

Thus, we represent a population of 20 individuals by PS and it is set up as follows: $PS = (PS_1,\ldots,PS_{20})$. We consider a Steady-State real-coded GA model which applies a crossover operator based on the use of fuzzy connectives [23] and a mutation random operator. The selection operator selects the best individuals (PSn) from the population in order to use them as parents for the new offspring. We use the following

parameters: Pc=1.0ᶜ Pm=0.05 and the percentage of the population to be replaced during each generation is 30%.

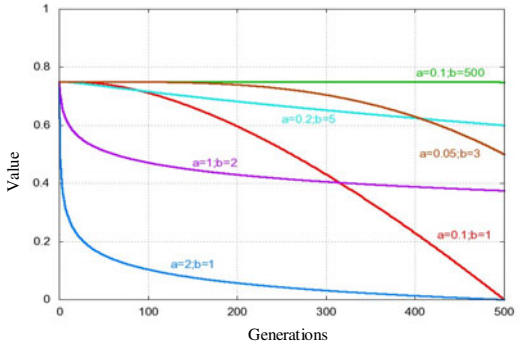

Fig. 5. Effect of the parameters a_i and c_i

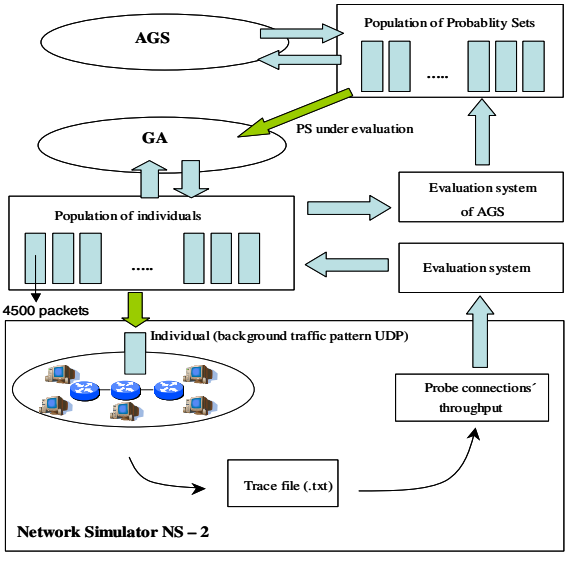

Fig. 6. Evolutionary learning of the PS of a GA

5 Results

The performance of the GA will be reviewed to drive the generation of a background traffic minimizing (critical background traffic) throughput in the computer network.

Standard statistical approach. For the standard statistical approach, where the background traffic is randomly generated according an equivalent Poisson process, we report the throughput obtained after simulating 30 times random patterns. As can see in Figure 7, the value does not change significantly, ≈ 1,5E+06 bits/sec, as new traffic patterns are randomly generated.

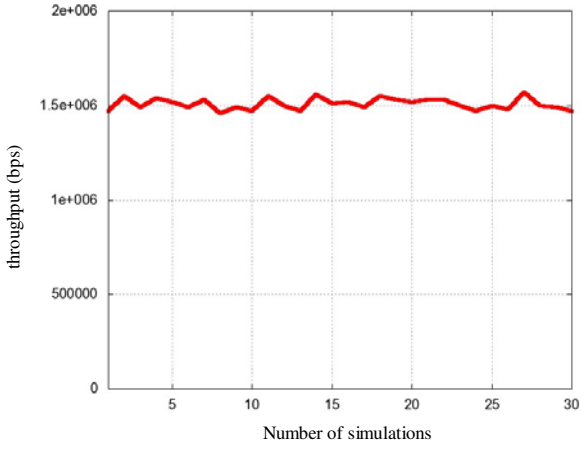

Fig. 7. Throughput of probe connections using Poisson's statistical processes

Using the static probabilities recommended. Table 3 show the results obtained when the GA uses the static probabilities recommend in [18] [19]: we have called GA-STATIC1 and GA_STATIC2 respectively. We run the GAs 30 times, each one with a maximum of 500 generations, in order to do later an experimental statistical analysis. The performance measures used are the following:

- **Average** performance: average of the lowest throughput obtained at the end of each run.
- **Generation** performance: number of generations after which improvements in solution quality were no longer obtained.

Table 3. Results obtained using the static probabilities recommended

Algorithm	Average (bps)	Generations
GA-STATIC1	655928	270
GA-STATIC2	674382	248

The lowest throughput obtained using GA-STATIC1 was 429382 bps and using GA-STATIC2 it was 502207 bps. In [18], the authors show a unique experiment and its result. The GA managed to degrade the probe connections´ throughput to 480000 bps.

Using an Adaptive Genetic Algorithm. An adaptive method presented in the literature [3] has been used, called GA-SELF (Self-adaptive control), which adapts the mutation probability during the run of the GA. We have considered the adaptation of this parameter since it can determine directly the degree of population diversity, which is the main factor to avoid the premature convergence problem [3]. The rest of the parameter values are the recommended in [4][19]. In the Self-adaptive control of Pm, an extra gen, *Pmi*, is added to the front of each individual and represents the mutation probability for all genes in the string. This gene evolves with the solution.

The values of *Pmi* are allowed to vary in the interval [0.001, 0.01] which has been chosen since it contains a wide spectrum of Pm values that were considered frequently in the literature. In this case, the average of the lowest throughput obtained at the end of each run has been 640813 bps, and the number of generations after which improvements in solution quality were no longer obtained was of 291. The lowest throughput obtained was 429382 bps 536255 bps.

Using the probabilities obtained from the AGS. When the AGS was introduced to the GA, the PS_n which obtained the highest mark was:

$$PS_{best} = \{0.7, 0.53, 0.0004, 0.38, 38.41, 0.75, 35.6, 0.57, 76.8\}$$

Later, the GA used these probabilities (in this case AGA) for each experiment. Again, we run the AGA 30 times each one with a maximum of 500 generations. The average of the lowest throughput obtained has been 438020 bps and the number of generations after which improvements were no longer obtained was of 347. The lowest throughput obtained was 304540 bps, as we can see in Figure 8.

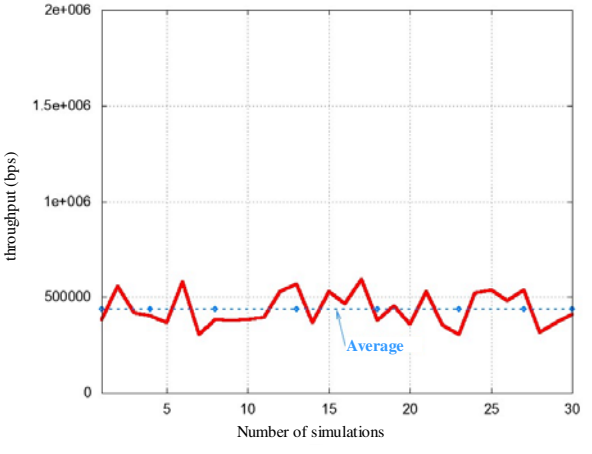

Fig. 8. Throughput using the probabilities obtained from the AGS

Statistical study. A t-student test was made in order to check if differences in the *Average* and *Generations* performance measures for AGS are significant when compared with the ones for the other algorithms. A significance level $\alpha = 0{,}05$ was applied for the test in question. Table 4 shows the results, where we can notice significant improvements in the average throughput using AGS. A plus sign (+) denotes an improvement in the performance, a minus sign (-) a reduction, and equal sign (=) non significant differences. As can be observed, the AGS has generated a background traffic pattern which significantly degrades the computer network performance.

Table 4. t-student test results

Algorithm	Average (bps)	Generations
GA-STATIC1	655928 +	270 -
GA-STATIC2	674382 +	248 -
GA-SELF	640813 +	291 =
AGS	438020	347

6 Conclusions

The paper presents an effective approach for control parameters optimization of a Genetic Algorithm, which has been applied to a real problem. The proposed system uses a meta-GA combined with an adaptation strategy of the GA control parameter. The results proved that, when the background traffic is driven by a GA which uses the probabilities obtained from the AGS, the computer network performance is much lower than when the traffic is generated by Poisson's statistical processes and by other algorithms. The AGS has identified the worst traffic pattern to the protocol. Using Poisson's statistical processes as models of background traffic is not possible to analyze the worst situation in a computer network. Poisson's statistical models are optimistic models to verify the computer network performance. The AGS is able to find realistic traffic loads to test the computer network. Finally, a two-sided t-test at 0,05 level of significance has been applied in order to ascertain the significant differences.

References

1. Goldberg, D.E.: Genetic Algorithms in search, optimization and Machine Learning. Addison-Wesley, New York (1989)
2. Cordón, O., Herrera, F., Hoffmann, F., Magdalena, L.: Genetic Fuzzy Systems: Evolutionary tuning and learning of fuzzy knowledge bases. Advances in fuzzy systems – Applications and theory, vol. 19. World Scientific Publishing, Singapore (2001)
3. Herrera, F., Lozano, M.: Fuzzy adaptive genetic algorithms: design, taxonomy, and future directions. Soft Computing 7, 545–562 (2003)
4. Grefenstette, J.J.: Optimization of control parameters for genetic algorithms. IEEE Transactions on Systems, Man and Cybernetics 16(1), 122–128 (1986)
5. Eiben, A.E., Michalewicz, Z., Schoenauer, M., Smith, J.E.: Parameter Control in Evolutionary Algorithms. In: Parameter Setting in Evolutionary Algorithms. Springer, Heidelberg (2007)
6. De Jong, K.: An analysis of the behaviour of a class of genetic adaptive systems. PhD thesis, University of Michigan (1975)
7. Grefenstette, J.J.: Optimization of control parameters for genetic algorithms. IEEE Transactions on Systems, Man and Cybernetics 16(1), 122–128 (1986)
8. Bramlette, M.F.: Initialization, mutation and selection methods in genetic algorithms for function optimization. In: Proceedings of the Fourth International Conference on Genetic Algorithms, pp. 100–107. Morgan Kaufmann, San Mateo (1991)
9. Eiben, A.E., Hinterding, R., Michalewicz, Z.: Parameter control in evolutionary algorithms. IEEE Transactions on Evolutionary Computation 3, 124–141 (1999)

10. Cicirello, V.A., Smith, S.F.: Modeling GA performance for control parameter optimization. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 235–242. Morgan Kaufmann Publishers, Las Vegas (2000)
11. Michalewicz, Z., Schmidt, M.: Parameter Control in Practice. In: Parameter Setting in Evolutionary Algorithms. Springer, Heidelberg (2007)
12. Wolpert, D.H., MacReady, W.G.: No free lunch theorems for optimization. IEEE Transactions on Evolutionary Computation 1(1), 67–82 (1997)
13. Hinterding, R., Michalewicz, Z., Eiben, G.: Adaptation in Evolutionary Computation: A Survey. In: Proc. of the IEEE Conference on Evolutionary Computation, pp. 65–69 (1997)
14. De Jong, K.: Parameter Setting in EAs: a 30 year Perspective. Parameter Setting in Evolutionary Algorithms. Springer, Heidelberg (2007)
15. Hui, J.: Switching and Traffic Theory for Integrated Broad Band Networks. Kluwer Academic Publisher, Dordrecht (1990)
16. Karagiannis, T., Molle, M., Faloutsos, Broido, A.: A nonstationary poisson view of internet traffic. Proc. of the Twenty-third Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM), Hong Kong, vol. 3, pp. 1558–1569 (2004)
17. Cao, J., Cleveland, W., Lin, D., Sun, D.: Internet traffic Tends Toward Poisson and Independent as the Load Increases. In: Holmes, C., Dennison, D., Hansen, M., Yu, B., Mallick, B. (eds.) Nonlinear Estimation and Classification 2002. LNS, pp. 83–109. Springer, Heidelberg (2003)
18. Corne, D.W., Oates, M.J., Smith, G.D.: Telecommunications Optimization: Heuristic and Adaptive Techniques. John Wiley and Sons Ltd., Chichester (2000)
19. Karthik, S., Jawajar, V., Chidambararajan, B., Srivatsa, S.K.: Performance of TCP over satellite networks under severe cross-traffic using GA. International Journal Mobile Communications 2(4), 382–394 (2004)
20. The Network Simulator -ns-2, `http://www.isi.edu/nsnam/ns`
21. Fernández-Prieto, J.A., Velasco, J.R.: Application of Genetic Algorithms in the research of the optimum probabilities of the genetic operators. In: 8th International Conference on Information Processing and Management of Uncertainty in Knowledge Based Systems (IPMU), pp. 291–297 (2000)
22. GAlib, `http://lancet.mit.edu/ga`
23. Herrera, F., Lozano, M., Verdegay, J.L.: The Use of Fuzzy Connectives to Design Real-Coded Genetic Algorithms. Mathware and Soft Computing 1(3), 239–251 (1995)