Adaptive Genetic Algorithm control parameter optimization to verify the network protocol performance

J.A. Fernández Prieto
Dpto. Ingeniería de Telecomunicación
Universidad de Jaén
Alfonso X El Sabio, 28
Linares (Jaén)
jan@ujaen.es

Juan R. Velasco Pérez
Dpto. de Automática
Universidad de Alcalá
Campus Universitario
Alcala de Henares (Madrid)
juanra@aut.uah.es

Abstract

Nowadays, it is important to test the computer networks under realistic traffic loads. One approach relies on integrating a Genetic Algorithm (GA) with the simulator of the system under verification. One of the main problems related to GA is to find the optimal control parameter values that it uses. Furthermore, different values may be necessary during the course of a run. Adaptive Genetic Algorithms (AGAs) have been built that dynamically adjust selected control parameters during the course of evolving a problem solution. In this paper we present a method of finding and dynamically adjusting the optimum probabilities to improve the GA performance and to drive the generation of a critical background traffic in a computer network.

Keywords: Adaptive genetic algorithms, background traffic, TCP, UDP.

1 Introduction

Nowadays, the telecommunication systems have reached a level of complication and complexity such that classic analytic approaches to their building and maintenance do not provide feasible solutions anymore. The computer networks are in continuous evolution and the protocols regulating their operation must be assessed for their suitability to new technologies. The effectiveness of a network solution depends on both the specifications of the protocols and their software and hardware implementations.

Protocols are sets of rules that govern the interaction of concurrent processes in distributed systems. The problem of designing efficient and unambiguous communication protocols existed long before the first computers were built. Entire networks can be paralyzed by faulty or incomplete protocols and therefore their specifications play a crucial role in the overall network performance. The protocols being developed today are very sophisticated offering more functionality and reliability, but as a result they have increased in size and complexity.

The problem that a designer now faces is how to design large sets of rules for information exchange that are minimal, logically consistent, complete and efficiently implemented. Thus, given a protocol, how can an analyzer demonstrate that it conforms to the correctness requirements? The analysis of the different aspects of communications protocols is usually done according to the following techniques [4]:

- Using Formal Description Techniques (FDTs), such as SDL, LOTOS or ESTELLE. They are methods to check that any protocol is correct and are powerful to capture most of the protocol semantics. However some aspects of a network protocol that are difficult to verify with these methods, for example, those

L. Magdalena, M. Ojeda-Aciego, J.L. Verdegay (eds): Proceedings of IPMU'08, pp. 785–791
Torremolinos (Málaga), June 22–27, 2008

related to different message queues, time-out counter, etc.

- Using simulation to verify the protocol. A simulator covers a very large number of applications, of protocols, of network types, of network elements and traffic models and it can explore the dependencies among the various hardware and software components of a computer network. The simulator models a number of sources in the network that can be programmed to generate traffic according to statistical models defined after real sources. One or more sources behave according to the protocol under study and the others are considered *background traffic* generators. Poisson statistical processes are often adopted as a model of background traffic.

In [4] the authors indicate that neither approach is sufficient to give enough information about the protocol performance. While the FDTs delivers a mathematically proven answer, it is forced to work on an over-simplified view of the protocol. On the other hand, the simulation rely on a clever choice of the traffic pattern to yield useful results, and worst-case analysis is often performed by hand-designing the traffic. They propose the adoption of a mixed simulation based technique that rely on a GA to explore the solution space and look for an inconsistency in the verification goal.

On the other hand, GAs use a number of parameters to control their evolutionary search for the solution to their given problems. Some of these include: population size, probability of crossover p_c, probability of mutation p_m, probability of selection p_s(to select the individuals from the population in order to use them as parents for the new offspring). The values of these parameters greatly determine whether the algorithm will find a near-optimum solution, and whether it will find such a solution efficiently. 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 the this experiment, we have used the following parameters: population size of 50, probability of crossover equal to 0.8, etc." without much justification of the choice made.

Finding robust control parameters setting is not a trivial task, since their interaction with GA performance is a complex relationship and the optimal one are problem-dependent [1]. 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, i.e. the need for good parameter tuning methods.

The problem of finding optimal control parameters for GAs has been studied by many: De Jong [6], Grefenstette [7], Schaffer [11], Bramlette [2], Wu - Chow [13], Eiben - Hinterding - Michalewicz [5], Cicirello - Smith [3], Herrera - Lozano [9], Subbu - Bonissone [12].

Furthermore, in [5] they arguments that any static set of parameters, having the values fixed during a GA run, seems to be inappropriate. Additionally, it is intuitively obvious that different values of parameters might be optimal at different stages of the evolutionary process. For instance, large mutation steps can be good in the early generations helping the exploration of the search space and small mutation steps might be needed in the late generations to help fine tuning the optimal individuals.

For these reasons, Adaptive Genetic Algorithms (AGAs) have been built that dynamically adjust selected control parameters 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 pa-

rameters can be quite difficult, designing an optimal function *p(t)* may be even more difficult.

In this paper we present a method of finding and dynamically adjusting the optimum probabilities to improve the GA performance and to drive the generation of a critical background traffic in a computer network. We compare its results with: a) the ones obtained in [4], where they carried out the experiments with a GA that used static probabilities (tuning "by hand"), and b) the ones obtained of a standard statistical approach (a common model for background traffic adopted in statistical network analysis is a Poisson process with exponentially distributed arrival times).

This paper is divided into 7 Sections, the first of which is the introduction we have just dealt with. Section 2 will show how to integrate a GA with a Network Simulator. Section 3 will present the components of the GA. Section 4 will show the topology of the network exploited in the experiments. Section 5 will describe the method to optimize the AGA control parameter. Section 6 will present the results obtained, and finally some conclusions in Section 7.

2 Integrating a GA with a Network Simulator

In [4] they show how to integrate a GA with a network simulator to drive the generation of a critical background traffic. The GA aims at generating the worst-case traffic for the protocol under analysis, given some constraints on the traffic bandwidth. Errors and weaknesses in the network protocol can there be discovered via simulation of this worst-case traffic. The figure 1 shows the 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 [10]. It is a discrete event simulator targeted at networking research and provides substantial support for simulation of TCP, routing, and

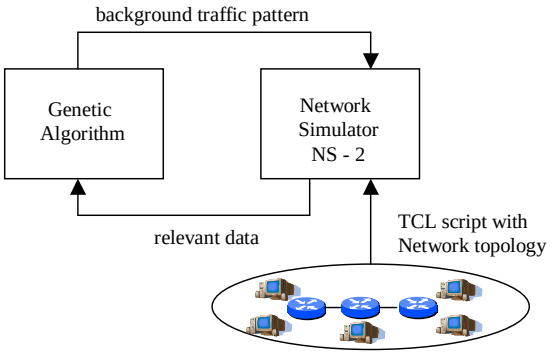

Figure 1: Architecture of the environment that integrates the GA and the simulator

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 explotation 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 that the background traffic made. Such information is then used to drive the generation of traffic patterns to be used in subsequent steps.

3 Genetic Algorithm

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

1. A genetic representation. 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.

2. Formation of an initial population. The background traffic corresponding to the initial population is generated according to a Poisson process whose interarrival time between packets is exponentially distributed.

3. Evaluation of individual fitness. 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. The fitness function should increase with the increasing goodness of a solution, and 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.

4. 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.

5. Values for the parameters that the GA uses. The parameters used by the GA In [4] are summarized in table 1.

But, are these static probabilities the optimal ones to obtain the best GA behavior?. Instead of using these probabilities, we propose to use other (that may change over time) that previously have been found with our method. The section 5 describes this method.

Table 1: GA parameter values

Parameter	**Value**
population size	50
selection probability	0.4
crossover probability	1.0
mutation probability	0.01
number of generations	500

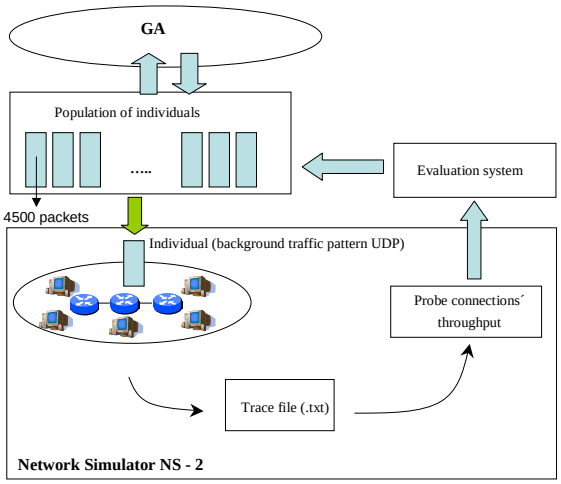

Figure 2: Genetic Algorithm

Some of the software for the GA used the GAlib genetic algorithm package, written by Matthew Wall at the Massachusetts Institute of Technology.

4 Network topology

The topology of the network exploited in the experiments is shown in figure 3. Three TCP connections span form the transmitters TX_i to the receivers RX_i 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 (BS_i) generate background UDP traffic directed to their respective destinations (BD_i) 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 10 μs 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.

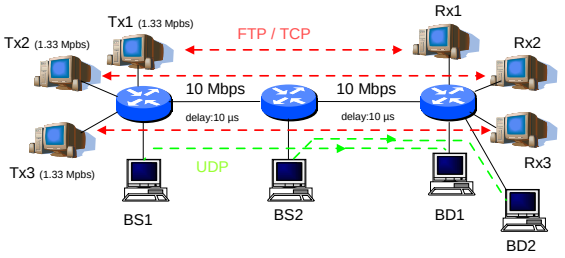

Figure 3: Topology of the network

5 Adaptive Genetic Algorithm control parameter optimization

The aim is to find the optimum probabilities to improve the GA performance. These probabilities are dynamically adjusted and changed over time. We have used three functions $p_i(t)$, $i = 1, 2, 3$ (t is the generation counter) where each one will affect, of independent form, to the P_s, P_c and P_m respectively. Each function will be of the following form:

$$p(t) = p_0^i\left(1 - \frac{Ln(t+1)^{(1/a_i)}}{(c_i Ln(T+1))^{(1/a_i)}}\right) \qquad \begin{matrix} a_i \in]0,2] \\ c_i \in [1,\mathrm{T}] \end{matrix} \quad (1)$$

where p_0^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 parameters a_i and c_i will model the concavity or convexity of the curvature and the attenuation of the initial value respectively. Figure 4 shows different curvatures depending on the values a_i and c_i for T=500 and $p_0^i = 0.75$.

In order to do the optimization of the probabilities that the GA uses, as well as of the parameters a_i and c_i of each function $p_i(t)$, we will apply an Additional Genetic System (AGS) to the GA, as is shown in the figure 5. In the population of the AGS a candidate solution is PS_n, n = 1,...,20, which represents

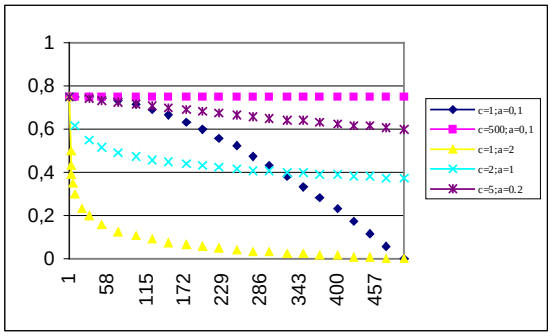

Figure 4: Effect of the parameters a_i and c_i

the set of values applied to the genetic operators which are used by the GA along with the parameters a_i and c_i. Each PS_n, codes a vector of real values in the following way:

$$PS_n = \{p_s, p_c, p_m, a_1, c_1, a_2, c_2, a_3, c_3\}$$

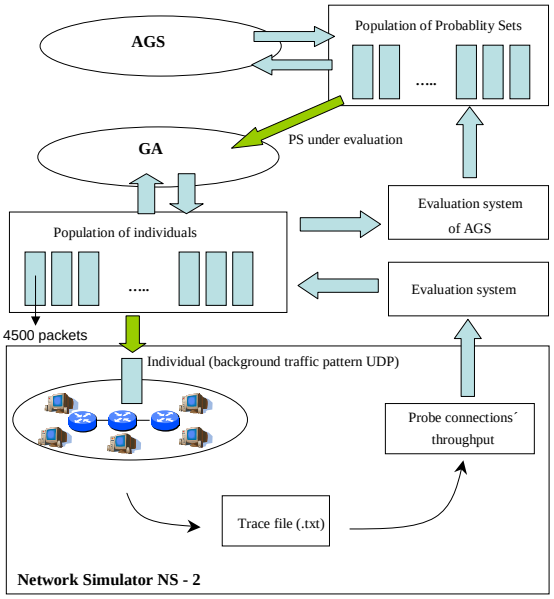

Figure 5: Evolutionary learning of the PS of a GA

Thus, we represent a population of 20 individuals by PS and it is set up as follows: PS = $(PS_1, \ldots, PS_{20})$. We have considered a steady-state GA model that applies a crossover operator based on the use of fuzzy connectives [8] and a mutation random operator. The selection operator selects the best individuals (PS_n) from the population in order to use them as parents for the new offspring.

The elitist strategy [6] is considered as well. We used the standard parameter settings for a population of 20 individuals [11].

6 Results

Now, we present the results. When the AGS is added to the AG, the the PS_n of the population PS obtaining the highest mark, was the one consisting of:

PS_{best} = { 0.70, 0.53, 0.0004, 0.38, 38.41, 0.75, 35.66, 0.57, 76.85}

Later, we will apply these probabilities in the GA (in this case AGA) for each experiment and all the conditions/assumptions are exactly the same as in [4]. We run the AGA 15 times, each one with a maximum of 500 generations. Table 2 shows the results obtained. The performance measures used are the following:

- **Min** performance: the lowest throughput (bits/seg) obtained at the end of each run.
- **Gen WL** performance: number of generations after which improvements in solution quality were no longer obtained.

Table 2: Throughput of probe connections obtained with the AGA.

Experiment	Min	Gen WL
1	**381148**	287
2	558008	232
3	**417087**	377
4	**401955**	120
5	**365070**	442
6	578815	470
7	**304540**	492
8	**382094**	426
9	**379256**	66
10	**383039**	498
11	**396280**	262
11	529635	336
13	568412	342
13	**366015**	156
15	528689	498

Figure 6 shows the throughput of the probe connections for the experiment 7. The values obtained are compared to the ones of a standard statistical approach, where the background traffic is randomly generated according an equivalent Poisson process. For the statistical approach, we report the lowest throughput obtained after simulating 15 times random patterns. This value does not change significantly as new traffic patterns are randomly generated. As it can see the AGA managed to degrade the probe connections´ throughput from 1.5 Mbps to 0.3 Mbps (in the best case).

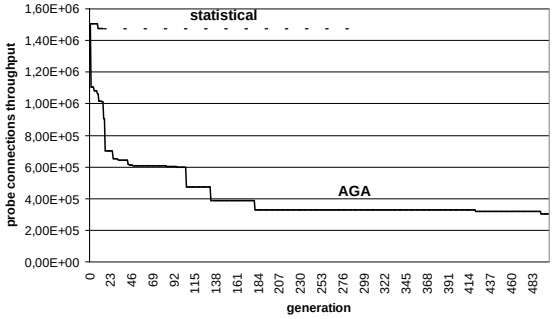

Figure 6: Throughput of the probe connections for the experiment 7

On the other hand, in [4] they carried out the experiments with a GA that used the probabilities shown in table 1. The GA managed to degrade the probe connections´ throughput to 0.48 Mbps. They show the result of an only experiment (we suppose that it is the best one obtained). Therefore, the AGA (using the probabilities obtained by the AGS) improves in 10 of the 15 experiments, that we have done, to the GA.

7 Conclusions

If we bear in mind the results of the experiments, we have to remark the following aspects:

- The behaviour of a GA is strongly determined by the probabilities used.
- Using the AGS is more time consuming however the application of the probabili-

ties obtained from the AGS in the AGA provides a better results.

- The AGA probabilities have been obtained for this particular problem and for this reason it is possible that we will get other results when dealing with other problems.

- The results proved that, when the background traffic is driven by the AGA, the TCP performance is much lower than when traffic is generated by statistical methods or when is driven by a GA that uses the probabilities shown in table 1.

References

[1] Bäck, T (1992). The interaction of mutation rate, selection, and self-adaptation within genetic algorithm. In *Männer R, Manderick B (eds), Parallel Problem Solving from Nature 2*, pp. 85–94. Amsterdam: Elsevier Science Publishers, 1992.

[2] Bramlette, M. F. (1991). Initialization, mutation and selection methods in genetic algorithms for function optimization. In *Belew, R. K., and Booker, L. B., eds., Proceedings of the Fourth International Conference on Genetic Algorithms*, 100-107. San Mateo, CA: Morgan Kaufmann, 1991.

[3] Cicirello, Vincent A. and Smith, Stephen F. (2000). Modeling GA performance for control parameter optimization. In *D. Whitley, D. Goldberg, E. Cantu-Paz, L. Spector, I. Parmee, and H. Beyer, editors, GECCO-2000: Proceedings of the Genetic and Evolutionary Computation Conference*, pages 235-242. Morgan Kaufmann Publishers, Las Vegas, July 2000.

[4] Corne, David W., Oates, Martin J., Smith, George D. (2000). Telecommunications Optimization: Heuristic and Adaptive Techniques. *John Wiley and Sons, Ltd. 2000.*

[5] Eiben, A. E., Hinterding, R., Michalewicz, Z. (1999). Parameter control in evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 3, pages 124-141, 1999.

[6] De Jong, K. (1975). An analysis of the behaviour of a class of genetic adaptive systems. *PhD thesis*, University of Michigan, 1975.

[7] Grefenstette, J.J. (1986). Optimization of control parameters for genetic algorithms. *IEEE Transactions on Systems, Man and Cybernetics*, 16, N 1, pages 122-128, 1986.

[8] Herrera, F., Lozano, M., Verdegay, J.L. (1995). The Use of Fuzzy Connectives to Design real-Coded Genetic Algorithms. *Mathware and SoftComputing.* Vol. 1:3, pp. 239-251, 1995.

[9] Herrera, F., Lozano, M. (2003). Fuzzy adaptive genetic algorithms: design, taxonomy, and future directions. *Soft Computing 7*, pp. 545–562, Springer-Verlag 2003.

[10] The Network Simulator -ns-2. http://www.isi.edu/nsnam/ns.

[11] Schaffer, J.D., Caruana, R.A., Eshelman,L.J. and Das, R. (1989). A study of control parameters affecting online performance of genetic algorithms for function optimization. *Proceedings of the Third International Conference on Genetic Algorithms and Their Applications*, San Mateo, California, pages 51-60, June 1989.

[12] Subbu, R., Bonissone, P. (2003). A retrospective view of Fuzzy Control of Evolutionary Algorithm resources. *Proceeding of the IEEE Conference on Fuzzy systems*, May 25-28, 2003, St. Louis, USA.

[13] Wu, S. J., and Chow, P. T. (1995). Genetic algorithms for nonlinear mixed discrete-integer optimization problems via meta-genetic parameter optimization. *Engineering Optimization* 24(2):137-159, 1995.