Genetic Algorithms and Cellular Automata: A New Architecture for Traffic Light Cycles Optimization

Javier J. Sánchez
Centro de Innovación para
la Sociedad de la Información
(C.I.C.E.I.)
University of Las Palmas de G. C.
35017-Las Palmas de Gran Canaria,
Spain
Email: jsanchez@polaris.ulpgc.es

Manuel Galán
Centro de Innovación para
la Sociedad de la Información
(C.I.C.E.I.)
University of Las Palmas de G. C.
35017-Las Palmas de Gran Canaria,
Spain
Email: manolo@cicei.com

Enrique Rubio
Centro de Innovación para
la Sociedad de la Información
(C.I.C.E.I.)
University of Las Palmas de G. C.
35017-Las Palmas de Gran Canaria,
Spain
Email: rubio@cicei.com

Abstract— **We present a new architecture for the optimization of Traffic Light Cycles in a Traffic Network. The model is based on three basic design items: The use of Genetic Algorithms as an optimization technique, the use of Cellular Automata Simulators within the Evaluation Function, and the use of a Beowulf Cluster as parallel execution environment for this architecture. We also present some tests to demonstrate that this new architecture is suitable for the problem it solves.**

I. Introduction

Nowadays traffic management is a first priority problem for many cities. Often extending traffic infrastructures is not viable and it is a very expensive solution. Moreover, environment impact is a very important matter we cannot forget.

It would be more convenient to optimize the existing infrastructures in order to get the best service from them, before new ones are constructed. For this reason, many research groups around the world are working on optimization methodologies for traffic.

Traditionally, the traffic management has been approached using the Trial-And-Error method: an expertise person or team decided on traffic parameters and depending on the resulting traffic behavior some feedback corrections occurred. This philosophy hasn't changed so much in past decades, except for the use of simulators instead of real traffic tests as feedback source. Recently some micro-simulators — based on the vision of traffic as a collection of independent vehicles — have proved to be very accurate.

Although the use of these tools has permitted the execution of many different simulations in relative short time,the method still depends on the engineer experience and "art", and moreover it can't ensure that the whole search space is covered.

Traffic optimization comprises a set of different problems, from which one of the most relevant ones is the traffic light cycles optimization. Traffic light cycles optimization is a NP-Complete problem. Considering that in these NP-Complete problems a deterministic optimizing method is not known, we find suitable to use non-deterministic optimizing methods like Genetic Algorithms.

The rest of this article is organized as follows. In the next subsection we comment on some other group efforts regarding traffic optimization. In section I-B we explain some concepts related to traffic simulation. In section II we formulate the problem. Section III describes our architecture in three steps, namely Genetic Algorithm description, Traffic Simulator description and Beowulf Cluster description. Section IV comments some of the test results we have obtained. Finally, section V includes our conclusions and some future work ideas.

A. Other Groups Efforts

Many research groups have dedicated their efforts to improve the use of traffic infrastructures. Some groups work on the optimization of traffic light cycles at a stand-alone intersection. For example in [1] every intersection is optimized considering only local information. Moreover, it can be adapted to short and long time traffic fluctuations.

Other groups work on the optimization of several intersections at the same time. An example is [2]. In this work an "ad hoc" architecture is used to optimize a 9 intersections traffic network. It uses Genetic Algorithms as an optimization technique running on a single machine. The CORSIM model is used within the evaluation function of the Genetic Algorithm. In this work scalability is not solved. Authors recognize that it is a customized non scalable system.

The offset time between two traffic lights is optimized with Artificial Neural Networks (ANNs) by [3]. The ANN is trained with the data provided from a simulator built up for this purpose. The ANN produces estimations of the length of queue of vehicles stopped in front of a traffic light. It allows the calculation of the optimal offset between two consecutive sets of traffic lights.

In [4] the stop and go traffic waves are minimized using an integral functional. They face it as a non standard problem of calculus of variations. Given a system of hyperbolic conservation laws, they introduce an integral functional.

Currently we optimize the traffic light cycles and the maximum speed allowed in a set of interconnected streets.

0-7803-8515-2/04/$20.00 ©2004 IEEE

In this work we present only the results of traffic light cycles optimization. We use a Cluster Beowulf as a cheap, scalable and flexible MIMD parallel computer.

B. Traffic Simulation

Traffic Simulation is known to be a very difficult task. There are two different sorts of traffic models. The first one is the macroscopic model set. They are based on Fluid Dynamics since they consider traffic flow as a continuous fluid. On the other hand, we have microscopic approaches. In them, traffic is considered as a set of discrete particles following some rules. In the last decade there is a common belief about the better performance of Microscopic approaches to do Traffic Modeling. One such widely used approach is the Cellular Automata Model.

Scientific literature is plenty of macroscopic approaches for traffic modeling. In the 50s appeared some "first order" continuum theories of highway traffic. In the 70s and later some other "second order" models were developed in order to correct the formers' deficiencies. References [5], [6], and [7] may illustrate some of these models. In [5] a continuous traffic model is proposed taking into account the safe distance each driver keeps. Moreover it extends the Navier-Stokes-like equations for the velocity variance of vehicles. In [6] the nonlinear theory of the cluster effect in a traffic flow, i.e., the effect of the appearance of a region of high density and low average velocity of vehicles is presented. In [7] it was presented a very well known free way simulation model. It formulated a variant of the equilibrium speed-density hypothesis that overcame the spontaneous lockup problem by adding a "look-ahead" term to the speed equation.

However, for example in [8] "second order" models are questioned due to some serious problems, i.e. a negative flow predictions and negative speeds under certain conditions.

Nowadays the microscopic simulators are widely used. Their main drawback is that they have a lower performance than the macroscopic ones, except in the case of the Cellular Automata. Cellular Automata and macroscopic simulators have similar computing times.

The Cellular Automata Simulators are based on the Cellular Automata Theory developed by John Von Neumann [9] at the end of the forties at the Logic of Computers Group of the University of Michigan. In this theory vehicles — in the traffic simulation case — are considered as discrete unidimensional entities. The streets are sampled into a set of points. There can be only one vehicle at each point. It establishes the rules over them and let them circulate freely. In the Cellular Automata model not only space is sampled into a set of points, but also time and speed are sampled too. Time becomes iterations. It sets a relationship between time and iterations, e. g. $1second \equiv 1iteration$. Consequently, speed turns into "points over iterations". If we sample a point every 2m and we have a vehicle moving at 10m/s (36Km/h) in our Simulator it will be represented as a vehicle moving at a 5 points per iteration speed.

In [10] we can find a well described list of microscopic models and a comparative study of them. Although conclusions are not definitive, this work seems to demonstrate that models using less parameters have better performance.

We have developed a traffic model based on the SK model ([11]) and the SchCh model ([12]). The SchCh model is a combination of a highway traffic model — Nagel-Schreckenberg [13] — and a very simple city traffic model — Biham-Middleton-Levine [14]. The SK model adds the "smooth braking". We decided to base our model in the SK model due to its better results for all the tests shown in [10].

The Nagel-Schreckenberg model simulates highway traffic. It consists of an one-dimensional lattice of cells that can accommodate no more than one vehicle at a time. Each vehicle is characterized by a maximum velocity and a randomization parameter. It considers the number of free cells in front and accelerates when possible to the maximum velocity or decelerates when necessary. An extra random parameter makes the vehicle to decelerate even if it is not necessary.

In the Biham-Middleton-Levine model a square lattice models the traffic network. Each cell represents a crossing of east-bound and north-bound streets. It can be empty or occupied by a vehicle moving to the north or to the east. At odd time steps only the east-bound vehicles are updated, and at the even time steps the north-bound vehicles are updated.

The SchCh model combines the Biham-Middleton-Levine model with the Nagel-Schreckenberg model rules for vehicle dynamics. The SK model is an extension of Nagel-Schreckenberg. It avoids discontinuous velocity changes.

In [15] it is demonstrated that the traffic light cycles have a strong influence in traffic flow results. This is the reason why we decided to deal with this problem.

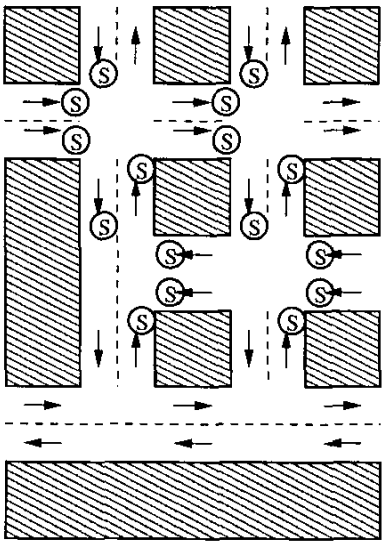

Fig. 1. Initial traffic topology

II. Problem Definition

We start from a traffic network like the one in figure 1. It consists of five two-lane streets. The symbol '*S*' means a traffic light. Circulation sense is indicated with arrows. For this network we try to find the optimal cycle of traffic light combinations during a fixed period — 'PERIOD_SIZE'. This cycle is repeated indefinitely.

The restrictions are the following:

1) There can only be a green traffic light by intersection every cycle step.
2) Too short traffic light transitions are forbidden. Every cycle step comprises a constant number of simulation iterations — 'Traffic light granularity'.
3) In our work we only consider two traffic light states namely green and red.
4) The input data for our optimization is set off-line, and is based on statistical information.

III. New Architecture Description

A. Genetic Algorithm Description

1) Chromosome Codification: In this work we optimize the traffic light cycles for all the intersections of a traffic network. Every cycle is represented by a word of PERIOD_SIZE integers. In every integer it is coded with two bits which traffic light is open at every cycle step. In figure 2 we show our chromosome.

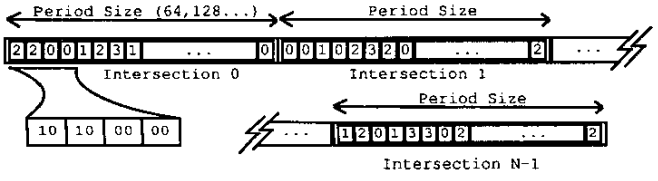

Fig. 2. Traffic Light State Chromosome

2) Selection, Crossover and Mutation Operators: We have chosen an Elitist Selection Strategy. It means that at every generation a little group of individuals — the best two individuals in our case — is cloned to the next generation. The rest of the next generation is created by means of crossover of their "parents" and mutation — those chosen by mutation probability.

Some other selection strategies like Tournament Selection and Probabilistic Tournament Selection have been tried. However, for our problem they seem to provoke premature convergence.

A standard two points crossover is used. We employ a variable mutation probability. This is because we believe that a high initial mutation probability would diversify the population and avoid local Minna. When a mutation occurs we proceed as follows. In the chosen individual we randomly chose the mutating gene position from all genes — and all intersections. Then we overwrite its value by a random value inside the rank of traffic lights at this intersection, i.e. [0,3].

3) Evaluation and Fitness: For the evaluation we use the mean time within the system. This is the elapsed time (iterations) since a new vehicle arrives to the network until it leaves. We do believe that this simple and intuitive parameter gives an appropriate picture of the network performance.

Although we have tested other parameters such as traffic homogeneity, queues' lengths, etc., the mean time within the network seems to be the best one, also producing quite robust results. We do not discard the employment of other statistical values, whenever we can register improvements.

Our optimization problem is a minimization problem. This is the reason why we calculated Fitness simply as the inverse of the evaluation value — Mean time at the system.

B. Cellular Automata Simulator

In our model these are the rules applied to all the vehicles.

1) A vehicle ought to accelerate up to the maximum speed allowed. If it has no obstacle in its way (other vehicle, or a red traffic light), it will accelerate at a pace of 1 point per iteration, every iteration.
2) If a vehicle can reach an occupied position, it will reduce its speed and will occupy the free position just behind the preceding.
3) If a vehicle has a red traffic light in front of, it will stop at it.
4) Smooth Braking: Once the vehicle position is updated, the vehicle speed is also updated. To do this, the number of free positions from current position forward is taken into account.
5) Multi-lane: When a vehicle in intending to move on, or to calculate the number of free positions forward on its way, it is permitted to consider positions on other parallel lanes.

Multi-lane moving is achieved by means of storing the set of possible next positions for every point in the scheme and for every input-output couple. For example, in figure 3 there is a vehicle in the path from point 24 to point 73. If it is in point 25 it may move to points 2 or 3.

In [16] a more complex multi-lane traffic set of rules is presented. However, our multi-lane scheme is simpler. Moreover it has a very interesting feature. Our scheme permits a vehicle to change to other lane twice of more times just in the same updating step. For example, if a vehicle has a speed of 2 points per iteration, it may change to other lane twice in the same simulator iteration. We believe that this is a more realistic behavior.

Our Cellular Automata simulator is not stochastic. So far, we avoid stochasticity. The position updating is done by means of a recursive function. It explores all the network dependencies and orders the updating sequence, so deadlocks are avoided. Hence, in every simulator iteration all the vehicles have the possibility of movement — if there is a free position or the car in front moves too.

Initial experiments showed that our Cellular Automata Simulator had a stochastic behavior. It is a very undesirable characteristic, because we need that our evaluation function returns

always approximately the same value for an individual in order to guide the Genetic Algorithm up to the better solution. In a second version of our simulator we have given it more realism, letting the vehicles more flexibility for movement in order to reduce the variance of results. This happened but it was not enough. This is the reason why we finally decided to eliminate the random behavior for good.

We realize that traffic flow is an intrinsic stochastic process. However, we need to obtain deterministic results from our simulator in order to guide the Genetic Algorithm. We do believe that our simulator, even if its behavior is not stochastic, gives us a very accurate idea of how well does a chromosome work, and permits us to compare it to others by means of a well defined fitness function.

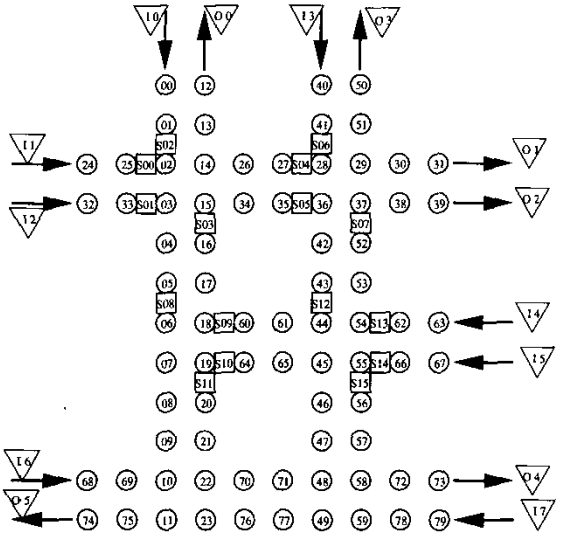

Fig. 3. Discretized traffic topology

C. Beowulf Cluster Description

Each time applications require more and more computing power. In the past, powerful systems were very expensive and thus prohibitive for most potential users. This situation has changed in the last years, since in 1994 T. Sterling and D. Becker developed a "distributed computing cluster" over Linux Operative System and using "Of The Shelf" hardware components. This new flexible, reusable and massively parallel sort of MIMD has a very good price/performance relationship, and is very suitable for intrinsic parallel tasks like the Genetic Programming ones.

The Architecture of our system is based on a five node Cluster Beowulf, due to its very interesting price/performance relationship, and the possibility of employing Open Software on it. On the other hand, this is a very scalable MIMD computer, a very desirable feature in order to solve all sort of traffic problems.

1) Hardware: Every cluster node consists of a Pentium IV processor at 3.06 GHz with 1 GB DDR RAM and 80GB HDD. Nodes are connected through a Gigabit Ethernet Backbone. Every node has the same hardware, except the master node having an extra Gigabit Ethernet network card for "out world" connection. The price of this hardware nowadays is about €5500.

2) Software: Every node has installed Red Hat 9 on it — Kernel 2.4.20-28.9, glibc ver. 2.3.2 and gcc ver. 3.3.2. It was also necessary for parallel programming the installation of LAM/MPI (LAM 6.5.8, MPI 2). For genetic programming, many Genetic Algorithms free libraries are available. However, in this work all the genetic algorithm architecture was developed in just plain ANSI C, only with the MPI parallel routines help.

In our application there are two kinds of processes, namely master and slave process. There is only one master process running on each test. In every generation it sends the chromosomes (MPI_Send) to slave processes, receives the evaluation results (MPI_Recv) and creates the next population. Slave processes are inside an endless loop, waiting to receive a new chromosome (MPI_Recv). Then they evaluate it and send the evaluation result (MPI_Send). The chromosome consists of a 64 bytes array when PERIOD_SIZE — The number of states in a traffic light cycle — is set to 64 steps, and 128 when it is set to 128 steps. The evaluation value is a double type (8 bytes).

TABLE I
M.E.T., SPEED-UP, AND B VARYING THE APPLICATION SCHEME

slave processes distribution	M.E.T.	S	B(%)	Std
Serial (N = 1)	1052.12	-	-	2.20551
N = 2 (1 slave/node)	533.65	1.9716	38.4012	1.56598
N = 3 (1 slave/node)	363.69	2.8929	18.2094	4.98604
N = 4 (1 slave/node)	272.21	3.8651	7.3405	0.61670
N = 5 (1 slave/node)	220.30	**4.7759**	**1.1731**	0.39385
N = 5 (2 slave/node)	324.21	3.2452	13.5186	1.08224
N = 5 (5 slave/node)	386.89	2.7194	20.9655	0.67207
N = 5 (10 slaves/node)	416.33	2.5271	24.4634	5.31319
N = 5 (20 slaves/node)	499.33	2.1046	34.3929	34.70410
N = 5 (40 slaves/node)	530.62	1.9828	38.0421	55.45181

IV. TESTS PERFORMED

We present three tests in this section. The first one is a Speed-up study in order to demonstrate that this is a useful architecture for parallelizing optimization tasks. In section IV-B we present one example optimization with some partial results and a fitness evolution picture. At last in section IV-C there is a comparison between the results of an optimized network and the results of other two non optimized systems.

The traffic network of our tests is shown in figure 1. In figure 3 we show the sampled scheme. We have considered approximately 7m. between two points — the minimal distance required in a traffic jam. In figure 3 we have represented

every traffic light with a rectangle, and inputs and outputs with triangles.

For all the tests performed there are some fixed values. The population size is 300 individuals. Every test goes on during 300 generations. We have used a variable mutation probability. It starts with a very high value (0.75) and decreases by a factor of 0.982. The relationship between time and iterations is $1iteration \equiv 1seconds$, more or less the human response time. And every simulation runs during 1500 iterations. For the tests in sections IV-A and IV-B PERIOD_SIZE is set to 64 and Traffic light granularity is set to 5 iterations. For test in sections IV-B and IV-C we chose the best Speed-Up application scheme we have found so far — 1 master and 5 slaves processes.

A. Parallel Speed-Up Study

In table I we represent the results. 10 executions have been used to calculate every mean value. In the first column we have the slave processes distribution. The input named *Serial* means only one master and one slave processes running over the same cluster node. It always runs one slave process per node except in the last five tests. In the second column the mean execution time (M.E.T.) for every application scheme is presented. The third and fourth columns represent the Speed-Up (S) and the serial percentage of our algorithm (B), both calculated following Amdahl's law (eq. 1). N means the number of processors. Finally the standard deviation of time results is shown.

$$S = \frac{T_{Serial}}{T_{Parallel}} = \frac{N}{(B * N) + (1 - B)} \quad (1)$$

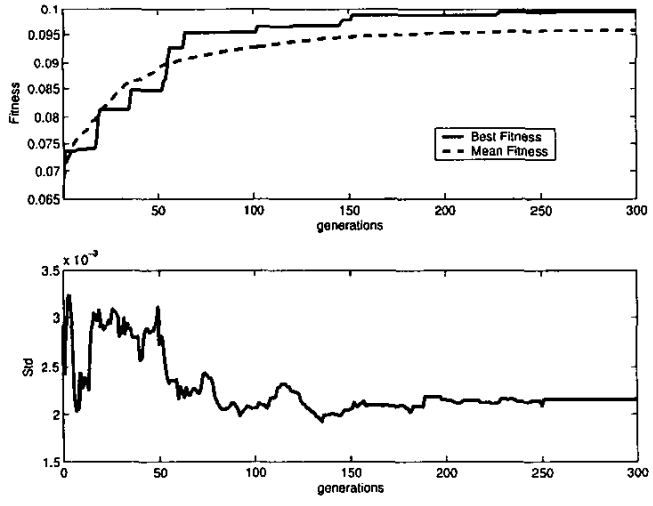

Fig. 4. Best Fitness and Average Fitness Evolution, and Standard Deviation

From table I two relevant discussion subjects emerge. First, one can notice a performance improvement as one increases the number of nodes sharing the work. The best situation is obtained when only one slave process is running at every node. For this situation there is a Speed-up of 4.7759 and B is 1.1731. It demonstrates the very high parallelism of genetic algorithm.

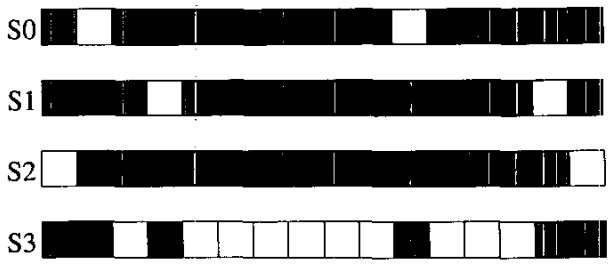

Fig. 5. Intersection 0 partial results

When we increase the number of slave processes at every node the performance — execution time — decreases. We believe that only one slave process per node permits the best computation and communication overlapping. When there are more than one slave per node there is a resource competition at every node that provokes not only a linear increase of slaves processing time, but also some context changing overhead. This seems to be the reason of the performance worsening.

B. Test Example Optimization

30 executions were performed for every PERIOD_SIZE and granularity couple. We made tests with values of 64 and 128 steps for PERIOD_SIZE and 5, 10, 20, 30, 40, 50 and 60 iterations for granularity. We present the results from the best test. It was achieved for PERIOD_SIZE 128 steps, and granularity 5 iterations. For this case the mean execution time is 1420.52s. We present the results from the best test. In figure 4 it is presented the best individual fitness evolution, all tests mean fitness evolution, and the Standard Deviation of them. The best fitness had an increase of 41.99%. The mean fitness had an increase of 39.60%.In figure 5 we displayed a fragment (16 steps) of traffic light cycles of one intersection. A black box means red light and a white box means green light. Every cycle step is equivalent to 5 simulation iterations — Traffic Light granularity — and consequently to 5 seconds.

The conclusion of this results is that we have a suitable genetic optimization in 300 generations.

C. Optimization Comparison

We have carried out several comparative studies to test how good are the optimized results in comparison to other simpler approaches. To do so we have carried out a set of tests with two different simulators. In the first simulator (RanSim) the traffic light cycles follow a random sequence.

In the second simulator (FixSim) there is a fixed green time in a fixed order for traffic lights. Every traffic light has the same green time as the rest.

For these two simulators and our system we have carried out two sets of tests, one setting the PERIOD_SIZE value to 64 steps (6), and one setting if to 128 steps (7). We do this to observe if there is any difference in results varying the traffic light cycle length.

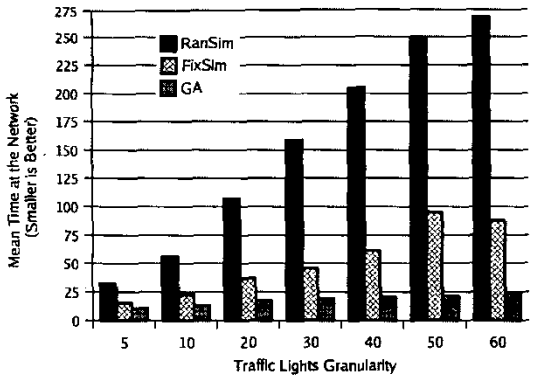

Fig. 6. Mean Times at Network (s) for PERIOD_SIZE=64

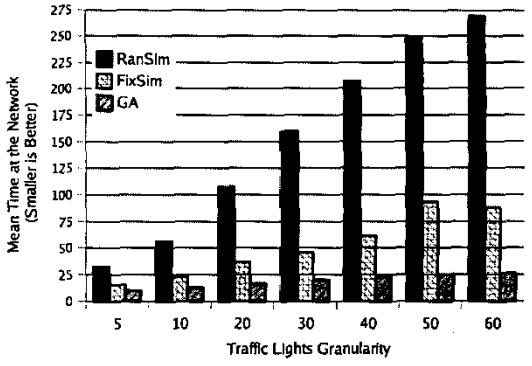

Fig. 7. Mean Times at Network (s) for PERIOD_SIZE=128

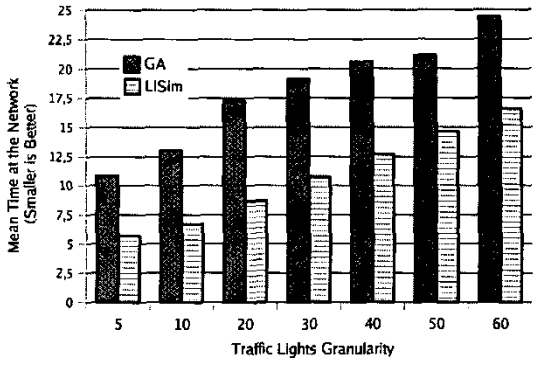

Fig. 8. Mean Times at Network (s) for PERIOD_SIZE=64

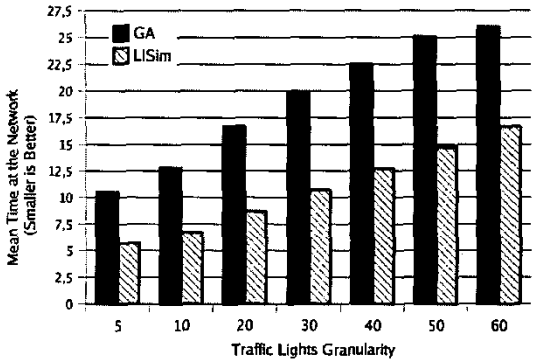

Fig. 9. Mean Times at Network (s) for PERIOD_SIZE=128

Additionally, we present other optimization results. This simulator (LISim) comprises a strong feedback and local intelligence system by means of a dynamic updating traffic lights state rule. The next green is decided depending on the number of occupied points behind the traffic lights at an intersection. The traffic light with the bigger number is chosen. If there is more than one traffic light with the same number of vehicles enqueued the next green is chosen randomly.

In figure 6 we present the mean time comparison for RanSim, FixSim and our optimized network setting the PERIOD_SIZE value to 64 steps. In figure 7 it is shown the mean time setting the PERIOD_SIZE value to 128 steps. 10000 simulations were used to calculate every mean time of FixSim and RanSim for every granularity value. Every result of our system is the mean value of 30 executions.

In figure 8 we present the comparison for LISim and our optimized network with PERIOD_SIZE equal to 64 steps. In figure 9 it is shown the same comparison setting the PERIOD_SIZE value to 128 steps. Again 10000 simulations were used to calculate every mean time of LISim and 30 executions for our system.

Some conclusions may be obtained observing these figures. In all cases there is a progressive worsening in results as we increase the traffic lights granularity.

In second place it seems that there is not a profound difference between results with a PERIOD_SIZE of 64 or 128 steps. The greater difference is a 6.5 % for the case of granularity equal to 60 with our system.

In third place, there is a strong improvement in our system from RanSim and FixSim. There is a mean decrease of a 83.82% comparing with RanSim and 56.74% comparing with FixSim.

Finally, from figures 8 and 9 we get a mean improvement of 42.78% in LISim against our system.

V. Conclusions and Future Work

We propose a new architecture, based on the combination of Genetic Algorithms over a MIMD computer — a Beowulf Cluster — using a Cellular Automata simulator within the evaluation function.

Up to date traffic network optimization has been faced using the Trial-And-Error method. This method cannot ensure that the whole searching space is covered. We propose a new method, a non deterministic optimization one to solve this task. In this work we optimize traffic light cycles. However, every traffic parameter — or set of parameters — may be

optimized following a similar procedure, and every scale of traffic network may also be optimized — scaling up hardware too.

Test IV-B demonstrates that this is a valid architecture for optimizing traffic light cycles. Moreover it is important to note that all intersections are optimized at the same time. We do believe that this sort of global optimization will produce better results than a stand-alone intersection optimization, such as [1].

The tests performed in IV-C are very interesting. It is demonstrated that our optimized traffic light cycles improve the results of other simpler approaches — FixSim and RanSim.

Secondly one may observe that results from LISim are much better than results from our architecture. LISim has a great advantage over our system. It is that it employs strong feedback information for the next green decision.

Frequently it is not possible to implement such a feedback system — like the one used by LISim — in a real life situation. The most of nowadays traffic networks are not yet prepared to give this sort of real-time information. So far the only information one can get in those systems is statistical information.

However we do believe that the best way to deal with traffic networks optimization goes through the implementation of systems using feedback information to correct its optimizations in real-time. This is the next step we are taking in our research.

Finally we want to note that our architecture has one additional benefit. It permits easily to multiply performance just scaling up hardware. Test results in section IV-A demonstrate this. This high parallelism is a very desirable feature for big networks or real time optimizations.

REFERENCES

[1] Vogel, A., Goerick, C., von Seelen, W.: Evolutionary Algorithms for Optimizing Traffic Signal Operation. Proceedings of ESIT 2000, Aachen, Germany (2000) 83–91

[2] Rouphail, N., Park, B., Sacks,J.: Direct Signal Timing Optimization: Strategy Development and Results. In XI Pan American Conference in Traffic and Transportation Engineering, Gramado, Brazil (2000)

[3] López, S., Hernández, P.,Hernández, A., and García, M.: Artificial Neural Networks as useful tools for the optimization of the relative offset between two consecutive sets of traffic lights LNCS. Foundations and Tools for Neural Modeling. Springer-Verlag. (1999) 795-804.

[4] Colombo, R. M., Groli, A.: Minimising Stop Go Waves to Optimise Traffic Flow. Appl. Math. Letters, To appear.

[5] Helbing, D.: An improved fluid dynamical model for vehicular traffic. Physical Review. **E51**, (1995) 3164.

[6] Kerner, B. S. and Konhäuser, P.: Structure and Parameters of Clusters in Traffic Flow Physical Review E,**50:54**, (1994)
%bibitemKuhne

[7] Payne, H. J.: Freflo: A macroscopic simulation model of freeway traffic. Transportation Research Record, 722:68, (1979).

[8] Daganzo,C.F.: Requiem for second order fluid approximations of traffic flow. Transportation Research B, (1995)

[9] von Neumann,J.: The General and Logical Theory of Automata. John von Neumann—Collected Works, Volume V, A. H. Taub (ed.), Macmillan, New York, (1963) 288–328.

[10] Brockfeld, E.; Khne, R. D.; Wagner, P.: Towards Benchmarking Microscopic Traffic Flow Models Networks for Mobility, International Symposium, Volume I, (2002) 321–331.

[11] Krauss, S., Wagner, P., and Gawron, C.: Metastable states in a microscopic model of traffic flow, Phys. Rev. E55 5597 – 5605 (1997)

[12] Schadschneider, A ; Chowdhury, D ; Brockfeld, E ; Klauck, K ; Santen, L ; Zittartz, J: A new cellular automata model for city traffic. "Traffic and Granular Flow '99: Social, Traffic, and Granular Dynamics". Springer, Berlin (1999)

[13] Nagel, K., Schreckenberg, M.: A Cellular Automaton Model for Freeway Traffic. Journal de Physique I France, **332** (1992) 2221–2229

[14] Biham, O., Middleton, A. A., Levine, D.: Phys. Rev. A **46** 6124 (1992)

[15] Brockfeld, E., Barlovic, R., Schadschneider, A., Schreckenberg, M.: Optimizing Traffic Lights in a Cellular Automaton Model for City Traffic http://arxiv.org/ps/cond-mat/0107056

[16] Wagner, P., Nagel, K. and Wolf, D. E.: Realistic Multi-Lane Traffic Rules for Cellular Automata. Los Alamos National Laboratory report LA-UR-96-2586, (1996)