
Javier Sánchez, Manuel Galán, Enrique Rubio
Centro de Innovación para la Sociedad de la Información (C.I.C.E.I.),
Universidad de Las Palmas de Gran Canaria,
Edificio Central del Parque Científico-Tecnológico,
Campus de Tafira, s/n.
35017-Las Palmas de Gran Canaria, Spain

Abstract. In a previous work we presented a new architecture for the optimization of Traffic Light Cycles in a Traffic Network. Three basic design items support the model: 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 a parallel execution environment for this architecture. In order to complete our work we present here a scalability study. We demonstrate clearly that this architecture is a very scalable one by showing the results of the optimization of four very different scale traffic networks.

Keywords: Traffic Lights Optimization, Genetic Algorithm, Cellular Automata, Traffic Simulator.

1 Introduction

One of nowadays cities major economical and social problem is traffic management. The traffic infrastructures get overloaded very quickly. Many times it is not viable to extend traffic infrastructures due to costs and lack of available space. Moreover, there is one more thing we cannot ignore: New infrastructures means new environment impacts.

Many traffic management initiatives provide on-line traffic information directed to drivers via Internet, Commercial Radio, GSM, electronic panels, etc. Some works have demonstrated that if drivers had this sort of information, it would suppose an improvement in traffic flow.

Work has mostly been targeted to solve concrete — and small — problems by “ad hoc” receipts. When more general — or wider — problems are faced, they are usually treated using the Trial-And-Error method: an expertise — person or team — decided on traffic parameters, and depending on the results some feedback corrections occurred.

For a long time this philosophy hasn’t changed too much. The use of simulators — micro-simulators have proven to be fast and accurate — instead of real traffic tests as a feedback source, implies that a great time saving tool is available. However, the method still depends on the engineer’s experience and ”art”. So it can’t ensure that the whole search space is covered.

Many other groups have dedicated much effort to optimize the existing infrastructures in order to get the optimal service from them before new ones are constructed. Some examples are presented in the next subsection. Some others were cited in (Sanchez et al. 2004).

One of the most relevant problems in traffic optimization is the traffic light cycles optimization. This is a NP-Complete problem so there isn’t a known deterministic optimizing method to solve it. Hence, it becomes necessary to use non-deterministic optimizing methods like Genetic Algorithms. In (Brockfeld et al. 2001) 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.

The rest of this article is organized as follows. In the next subsection, we comment on some other group efforts regarding traffic optimization. In 1.2 we explain some concepts related to traffic simulation. In section 2 we formulate the problem. Section 3 describes our architecture in three steps, namely Genetic Algorithm description, Traffic Simulator description and Beowulf Cluster description. Section 4 comments the test results we have obtained in order to perform the scalability study. Finally, section 5 includes our conclusions and some future work ideas.
1.1 Related Work

A good example of related work is (You Sik et al. 1999). The paper proposes the concept of the optimal green time algorithm, which reduces average vehicle waiting time while improving average vehicle speed using fuzzy rules and neural networks. Through computer simulation, this method has been proven to be much more efficient than fixed time interval signals. The fuzzy neural network will consistently improve average waiting time, vehicle speed, and fuel consumption. This work only considers a very small amount of traffic signals — two near intersections — in the cycle optimization.

In (Wann-Ming 2001), a cycle-based isolated intersection is controlled applying efficient optimal control techniques based on linear systems control theory to solve the linear traffic model problem. The main contribution of this research is the development of a methodology for alleviating the recurrent isolated intersection congestion caused by high transportation demand using existing technology. Again this work deals with very small scale traffic networks — one intersection.

In (Spall et al. 1994) the authors presented a neural network approach for optimal light timing. This paper introduces a fundamentally different approach that eliminates the need for an open-loop model. The approach is based on a neural network (or other function approximator) serving as the basis for the control law, with the weight estimation occurring in closed-loop mode via the simultaneous perturbation stochastic approximation (SPSA) algorithm. Since the SPSA algorithm requires only loss function measurements (no gradients of the loss function), there is no open-loop model required for the weight estimation. The approach is illustrated by simulation on a six-intersection network with moderate congestion and stochastic, nonlinear effects. A neural network is employed to implement the traffic lights control function. The training process of the NN is fed exclusively with real data. So, it would only be useful in systems with an on-line data acquisition module installed. However, up to date such systems are not common at all.

Finally, we cite (Di Febbraro et al. 2002). In this paper, Petri nets are applied to provide a modular representation of urban traffic systems regulated by signalized intersections. The basic idea is to consider such systems to be composed of elementary structural components, namely, intersections and road stretches. In order to describe the movement of vehicles in the traffic network, a microscopic representation is adopted and worked upon via timed Petri nets. An interesting feature of this model consists in the possibility of representing the offsets among different traffic light cycles as embedded in the structure of the model itself. Even though it is a very interesting work, the authors only optimize the coordination among traffic light timings. Our cycle optimization is a complete flexible one because we implicitly optimize not only traffic light offsets but also green times.

1.2 Traffic Simulation

![Figure 1. Small Traffic Network](image1)

![Figure 2. Small Discretized Traffic Network](image2)

There are mainly two different flavors of Traffic Simulation models, the macroscopic one and the microscopic one. The first one is based on Fluid Dynamics while the second one is based on discrete mathematical models like Cellular Automata. In (Sanchez et al. 2004) we comment on some examples of both of these
models. As we explained in that work, we have developed a traffic model based on the SK model ((Krauss et al. 1997)) and the SchCh model ((Schadschneider et al. 1999)). The SchCh model is a combination of a highway traffic model — Nagel-Schreckenberg model(Nagel et al. 1992) — and a very simple city traffic model — Biham-Middleton-Levine model (Biham et al. 1992). 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 (Brockfeld et al. 2002).

Nowadays the microscopic simulators are widely used. Their main drawback is that they have a lower performance than their macroscopic counterparts, with a sole exception: the Cellular Automata — Cellular Automata and macroscopic simulators have similar computing times. The main advantage of using Cellular Automata is that they seem to induce a more realist behavior.

![Traffic Light State Chromosome](image)

**Figure 3. Traffic Light State Chromosome**

## 2 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. The circulation sense is indicated with arrows. We try to find the optimal cycle of traffic lights 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.

## 3 Architecture

### 3.1 Genetic Algorithm Description

#### 3.1.1 Chromosome Codification

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, each one coded with two bits. Every integer indicates which traffic light is open at every cycle step for every intersection. In figure 3 we depict the structure of our chromosome.

#### 3.1.2 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 remainder of the next generation is created by means of crossover of their “parents” and mutate some of them — 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. We do not discard the use of other Selection Strategy once proven to be better.

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 minima.

The mutation works as follows. We randomly select the mutating gene in the chosen individual from all intersections. Then, we overwrite its value with a random value inside the rank of traffic lights at the respective
3.1.3 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 calculated Fitness simply as the inverse of the evaluation value since this is a minimization problem.

3.2 Cellular Automata Simulator

The Cellular Automata Simulators are based on the Cellular Automata Theory developed by John Von Neumann (Neumann 1963). 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 lets 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. Consequently, speed turns into "points over iterations". 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.
4. Smooth Braking: Once the vehicle position is updated, then 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 is trying to move on, or update its speed, it is allowed 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. Lets take Figure 2 as an example. If there were a vehicle in point 25, in the path from point 24 to point 73, it would be able to move to points 2 or 3.

Our Cellular Automata simulator is not stochastic. So far, we avoid stochasticity. The position updating is done through a recursive function. It explores all the network dependencies and orders the updating sequence, so deadlocks are avoided.

We realize that the traffic flow is an intrinsic stochastic process. However, we need to obtain deterministic results from our simulator to properly 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, permitting us to compare it to others by means of a well defined fitness function.
3.3 Beowulf Cluster Description

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 — and scales — of traffic problems.

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.

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).

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. The evaluation value is a double type (8 bytes).
Table 1. Four networks data. “Points” means the number of points resulting when sampling the respective network at a rate of a sample every 7 m approximately—the minimal distance required in a traffic jam. “T.Lights” means the number of traffic lights optimized. “Intersections”, “Inputs” and “Outputs” mean the number of intersections, inputs and outputs of the respective network. Finally “Chromosome Size” means the number of bytes that every chromosome includes, and “Initial Data Size” means the total amount of initial data broadcast to every slave before the optimization process starts.

<table>
<thead>
<tr>
<th>Scale</th>
<th>Points</th>
<th>T. Lights</th>
<th>Intersections</th>
<th>Inputs</th>
<th>Outputs</th>
<th>Chromosome Size</th>
<th>Initial Data Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Small</td>
<td>80</td>
<td>16</td>
<td>4</td>
<td>6</td>
<td>8</td>
<td>96 bytes</td>
<td>239.9102 Kbytes</td>
</tr>
<tr>
<td>Big</td>
<td>202</td>
<td>24</td>
<td>12</td>
<td>14</td>
<td>14</td>
<td>192 bytes</td>
<td>1.0150 Mbytes</td>
</tr>
<tr>
<td>Huge</td>
<td>248</td>
<td>40</td>
<td>20</td>
<td>18</td>
<td>18</td>
<td>320 bytes</td>
<td>2.0535 Mbytes</td>
</tr>
<tr>
<td>Immense</td>
<td>1176</td>
<td>160</td>
<td>80</td>
<td>36</td>
<td>36</td>
<td>640 bytes</td>
<td>24.3462 Mbytes</td>
</tr>
</tbody>
</table>

4 Scalability Study

In this section we present the scalability study. We have performed tests with four different scale networks. In figures 2 (small network), 4 (big network), 5 (huge network), 6 (immense network), we present them. In all figures we have represented every traffic light with a rectangle, and inputs and outputs with triangles.

In Table 1 we present the statistics of the networks.

![Fitness Evolution Graphs]

For all the tests performed there are some fixed values. The population size is 200 individuals. Every test goes on during 200 generations. We have used a variable mutation probability. It starts with a very high value (0.75) and decreases by a factor of 0.975. The relationship between time and iterations is $t_{\text{iteration}} = ...$
1 second, more or less the human response time. And every simulation runs during 1500 iterations.
PERIOD_SIZE is fixed to 64 steps and traffic light granularity is set to 5 iterations.

4.1 Fitness Evolution

For every network scale we ran 30 executions. We plot here three functions:

- Best Fitness Evolution: The best fitness value evolution of the best test.
- Mean Fitness Evolution: The average of all tests best fitness values evolution.
- Standard Deviation: The standard deviation of all tests best fitness value evolution.

Figure 7 presents these plots for every network scale. Figure 8 shows the achieved fitness improvement for every scale. In this figure two bars represent the best test fitness improvement and the mean improvement respectively.

In figure 9 we displayed a fragment (16 steps) of traffic light cycles of one intersection for the small scale case. A black box means red light while 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 these results is that we have a suitable genetic optimization in 200 generations for every scale. The mean fitness improvement keeps always over the 30%.

4.2 Execution Time Study

The objective of this subsection is to watch the architecture performance as the network scale gradually grows up. In Figure 10 we display the absolute mean Elapsed Time, and the ratios of the Elapsed Time to the population size, to the chromosome size and to the product of the population size and the chromosome size.

In this figure one can see that the relationship between scale and elapsed time keeps always stable, except for the last network scale. In this case the architecture seems to be overloaded. Hence, for “immense” scale it would be necessary to add an extension to the architecture to tackle it more efficiently.
4.3 Optimization Comparison

Finally, we have carried out many tests to know how good are the optimized results in comparison to other approaches. To do so we have implemented two different philosophies for traffic light state assignment. In the first simulator (RanSim) the traffic light cycle follows 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.

In figure 11 we present the mean time at every network scale for the RanSim case, the FixSim case and our optimized values. 10000 simulations were used to calculate every mean time of FixSim and RanSim, and the results of our system were found by calculating the mean value of 30 executions.

Additionally, we present other simulator results. This simulator (LISim) comprises a strong feedback and local intelligence system. The simulator performs a dynamic updating of the traffic light state. The next green is decided depending on the number of occupied points behind every traffic light 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 12 we present the comparison of LISim and our architecture for every scale. Again 10000 simu-
lations were used to calculate every mean time of LISim and 30 executions for our system.

Some conclusions may be obtained observing these figures. In figure 11 one may see that there is a strong improvement from our system to RanSim and FixSim, independently of the scale. From Figure 12 we observe that LISim only seems to be clearly better than our system for the “small” scale network. For the rest of the scales, the mean time values seem to be better in our system than in the LISim.

There is one thing more to say about figures 11 and 12. Mean times are smaller for the “immense” case than for the “huge” case. It seems senseless. But the sense of this effect is very simple. The number of simulation iterations (1500) is not enough to take into account largest paths. Only the shortest paths are taken into account when computing the mean values.

Hence, this is a second reason for the needing of a cluster extension.

5 Conclusions and Future Work Ideas

We proposed a new architecture in (Sanchez et al. 2004) based on the combination of Genetic Algorithms over a MIMD computer — a Beowulf Cluster — and a Cellular Automata simulator within the evaluation function. Thus far, traffic network optimization — traffic light timings optimization, in particular — has been basically faced using the Trial-And-Error method. This method cannot guarantee the whole searching space is being covered. We proposed a new method, a non deterministic optimization one, to solve this task.

With this paper we complement the former one with a scalability study. It demonstrates that this architecture is such a very scalable one. This is a very desirable feature because traffic optimization problems need very flexible platforms more than “ad hoc” solutions.

Tests on subsection 4.1 demonstrate that this is a valid architecture for optimizing traffic light cycles in a wide range of network scales.

The subsection 4.2 deals with the execution time versus network scale in order to prove that the architecture copes very well with a wide range of scales. We show that only in the case of “immense” scale there is an overload of architecture. This problem would be easily solved just scaling the hardware.

The tests performed in 4.3 are very interesting. In figure 11 it is demonstrated that our system improves on the results of other simpler approaches — FixSim and RanSim.

It is important to note that in our architecture 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 in (Vogel et al. 2000).

One may observe that the results from LISim — figure 12 — are much better than the ones from our architecture for the small scale case. LISim has an advantage over our system. Namely, it employs a strong feedback information system for the next green decision. However, as we increase the number of intersections and traffic lights optimized it becomes apparent that it is better to use a global optimization system than a local one. We pretend to research more in this direction.

We are still believing that the best way to deal with traffic networks optimization goes through the implementation of systems using feedback information to correct its global optimizations in real-time. This is a promising research line we have also aimed at.
References


