International Journal of Traffic and Transportation Engineering 2015, 4(1): 16-34
DOI: 10.5923/j.ijtte.20150401.03

AETROS: Dynamic Travel Route Assignment Coping with Highly Changeable Scenarios at Traffic Networks

Javier Sanchez-Medina[1,*], Enrique Rubio-Royo[1], Alexander Paz[2]

[1]Innovation Center for the Information Society (CICEI), Department of Computer Science, University of Las Palmas de Gran Canaria, Spain
[2]Department of Civil and Environmental Engineering, Director, Transportation Research Center, University of Nevada, Las Vegas

Abstract Dynamic Traffic Assignment is currently a key topic for the most advanced traffic management systems. Basically, it consists on automatically diverting traffic to the optimal choice of a set of alternatives – or combination of alternatives – towards each vehicle's destination node. A truly dynamic traffic assignment system could, potentially, cope with a highly variable traffic user demand and conditions, even with unexpected situations, like accidents. The research question for us is: Can we create an adaptive predictive architecture that provides, for each driver, the best choice at each intersection, for its destination and the current global traffic situation? In this work we share the results of an intensive set of tests accomplished with the Adaptive Evolutionary Travel Route System (AETROS). This model was designed to solve the travel route assignment problem in a new dynamic way. Urban traffic use to be highly changing scenarios where statistic based optimization approaches cannot cope with unforeseeable situations. When it is about assigning travel routes to users, it is usually approached in a static manner: when a traveler is assigned a route, it remains fixed from origin to destination. However, in the meantime the traffic scenario may suddenly change. And all those already assigned routes may became easily suboptimal. That is why AETROS is designed to be constantly adapting to the current situation and drivers are not assigned a fixed origin-destination route. Instead, as they approach an intersection they will receive the last optimal option for their destination. We have tested two different networks using a Beowulf cluster and a supercomputing facility. The smaller network was used to test whether the system correctly suggests the best option, by us introducing controlled accidents. The second bigger network has been used to deeply test the system through a wider range of traffic demands and accident probabilities. With the bigger network tests are subdivided into three different subsets. The same setup parameters where tested allowing drivers to use or not the suggested best options, forcing drivers to take the optimized options and finally without any route optimization. The presented results are meaningful, giving us important clues to keep on pushing the AETROS development.

Keywords Traffic Microsimulation, AETROS, Dynamic Travel Route Assignment, Cellular Automata, Parallel Genetic Algorithms

1. Introduction

Travel Route Assignment is a very important issue when planning, designing, and management of Intelligent Transportation Systems. Essentially, traffic is unpredictable. One can try to extract statistics for daily demand profiles depending on the day of the week, the labor year period, school vacations, local festive times, etc. However, when a person is behind the wheel of an vehicle, which is imperfect as all devices are – and this device has a large mass capable of moving at high speeds, determining traffic patterns that will never be predictable. Unforeseen events – such as accidents, demonstrations, and the effect of natural phenomena – can make even the most efficient traffic network break down. Therefore, there always will be a gap between a statistics-based approach and real traffic behavior, likely resulting in sub-optimal setups of the existing traffic networks.

By using a model-based predictive-control approach, the aim of this study was to generate travel route assignments that could adapt to quickly changing demand profiles [1]. In developing the optimization model, three objectives were considered. First, it was necessary to have an accurate picture of the current traffic situation. Second, the near-future forecast as to what the traffic situation might be needed to be trustworthy. Finally, the optimization system needed to be powerful and flexible in order produce the setup changes needed in the network as fast as possible so that its performance was maximized.

In order to develop this model, the current state-of-the-art methods in image detection were investigated as well the

* Corresponding author:
javier.sanchez@ulpgc.es (Javier Sanchez-Medina)
Published online at http://journal.sapub.org/ijtte

widespread availability of positioning technologies to assess the current traffic situation for the network in question. Regarding forecasting of future traffic, a cellular automata-based simulation model was used (see Section 3) that was very flexible and quite lightweight. Finally, to optimize the traffic network, a genetic algorithm (GA) running over a Beowulf cluster multi-computer was used because it is powerful, flexible, extendible, and has a very good price/performance ratio. The main idea of the model – called the Adaptive Evolutionary Travel Route Optimization System (AETROS) – was to provide drivers with a freshly optimized set of 'next options' as they approached an intersection; depending on their destination, these options were designed to maximize the system performance. Therefore, the approach used employed a system-optimal moving horizon.

This paper discusses the results of two sets of tests. The first test involved a small network having only two options in order to see whether the genetic algorithm worked reasonably well. The second one, a bigger and more complex network, served as a test bench to explore the strong and weak points of the system in order to determine the next direction the research should take.

This article is organized as follows. For the remainder of Section 1, past relevant research is discussed. Section 2 describes the overall elements of the AETROS model. In Sections 3 and 4, two components of the AETROS model that are of importance are explained in detail: the microscopic simulator and the genetic algorithm that was used. Finally, in Sections 5 and 6, the experimental results obtained are discussed, concluding remarks are in Section 7.

A small collection of related works was examined regarding 1) optimization of traffic lights using genetic algorithms and 2) parallel computing using a static statistic approach (e. g., [2], [3]). Similarly, studies on cellular automata-based traffic microsimulation were explored. In a beautiful work, Sheu [4] proposed an analytical solution to best-path calculation in the event of a lane-blocking incident. This author presented a step-by-step methodology to cope with such situations. Moreover, a set of numerical examples were performed to obtain the best results for conditions involving low-volume traffic flow. Gao and Chabini [5] proposed a policy-based stochastic model for dynamic traffic assignment (DTA). In this model, drivers do not have a predetermined path; instead, they make a decision at a set of intersections. The information used to make that decision is provided by the system, based on the vehicles that already have driven through that link. The drivers receive information close to the current travel time, more or less, by choosing one path or another.

Similar to the DRACULA microsimulation model developed at the University of Leeds [needs citation], Liu et al. [6] proposed a model that integrated drivers' decisions and vehicle movements by representing the drivers' day-to-day choices plus a detailed within-day traffic simulation model of vehicle trajectories according to car-following and lane-changing rules and intersection regulations. The paper described a modelling approach to road network traffic that emphasized the integrated microsimulation of decisions made by individual trip-makers and movements of individual vehicles across the network. To achieve this, the simulated represented directly the choices of individual drivers and their experiences as evolved from day to day, combined with a detailed within-day traffic simulation model of the space-time trajectories of individual vehicles. Therefore, it modeled both day-to-day and within-day variability in both demand and supply conditions. This is particularly suitable for realistic modelling of real-time strategies, such as those listed above. Full model specifications were given along with details of the algorithmic implementations. An impressive number of representative numerical applications were presented, including sensitivity studies of the impact of day-to-day variability, an application to the evaluation of alternative signal-control policies, and the evaluation of introducing bus-only lanes in a sub-network of Leeds.

Park and Kim [7] proposed a hybrid real-time and off-line approach to the dynamic path assignment problem. They used autonomous agents and systolic parallel processing as simulation paradigms. A repertoire of predetermined paths was computed off-line, using conventional optimal path-finding algorithms, such as the Frank-Wolf algorithm.

Mahut et al. [8] evaluated measures for on-line intelligent transportation system (ITS) measures, such as adaptive route-guidance and traffic management systems, which depend heavily on the use of faster-than-real-time traffic simulation models. Fast-running traffic models are needed off-line for use in iterative approaches that imitate drivers' adaptations to changes to the network topology. The paper describes a simulation-based dynamic-equilibrium traffic assignment model. The determination of time-dependent path flows was modelled as a master problem that is solved using the method of successive averages (MSA). The determination of path travel times for a given set of path flows is the network-loading sub-problem, which is solved using a space-time queueing approach of Mahut (2000). This loading method has been shown to provide reasonably accurate results with very little computational effort. The model was applied to the Stockholm road network, which at the time consisted of 4342 links, 1980 nodes, and 250 zones, representing over 11,000 turns. The results showed that this model was applicable to medium-sized networks with a very reasonable computation time.

Semi-Dynamic traffic route assignment approaches are quite common in literature, like [9]. That method proposed a very interesting multimodal methodology. However, it does not implement a fully dynamic route assignment, but it divides day in a number of periods, 15 to 90 min long. This is not as flexible as the method proposed in this paper, which can cope with really sudden conditions changes.

Another interesting work on DTA is presented in [10]. Their proposed methodology is challenging.

However, it relies in an assumption that may prove quite insufficient. They assume that travelers select their routes

based on their subjective better route perception. Our method, on the other hand does a global optimization.

In this current study, a stochastic approach is proposed to cope indirectly with lane-blocking incidents by means of a new dynamic model for travel-route optimization. The approach that Gao and Chabini [5] used is similar the model developed in this study. In both cases, drivers do not have a predetermined path; instead, they make decisions at a set of intersections. In the model developed in this study, drivers will not receive that kind of information to make their choices. Rather, they will receive the system-preferred option obtained by a global optimization using a genetic algorithm and by traffic microsimulation. The model developed by Liu [6] was found to be computationally feasible as a method for providing a fully internally consistent, microscopic, dynamic assignment, incorporating both within- and between-day demand and supply dynamics. This is an example of many research initiatives that implement prefixed paths for the traffic-route-assignment problem. In the case of this study, however, that kind of repertoire of possible paths does not exist. However, this model provides a set of possible options at the intersections of the traffic network provided to the drivers by an automatic terminal information service (ATIS), thus allowing the optimized best option depending on their destinations.

2. Travel Route Assignment Architecture

An urban traffic network is composed of lanes, intersections, and input lanes by which cars get into the network toward a set of destinations (outputs). This study tried to solve the challenge of how this can be best managed and to optimize traffic network performance without huge money investments. The traffic control model proposed does not need a big budget, and can improve traffic network performance.

The overall concepts of the AETROS model are summarized in Figure 1. For this research, the real traffic network was simulated. In near future, however, plans include obtaining sample data from an actual urban traffic network in order to have an accurate snapshot of the situation. That snapshot, whether simulated or real, must include the current position and speed of every vehicle in the zone in question. Additionally, virtual/actual drivers must communicate their destinations when entering into the network for two reasons: 1) to determine the destination of every new vehicle is as it arrives every optimized intersection and 2) to calculate the current Origin-Destination probability matrix in order to ensure that the simulations are accurate.

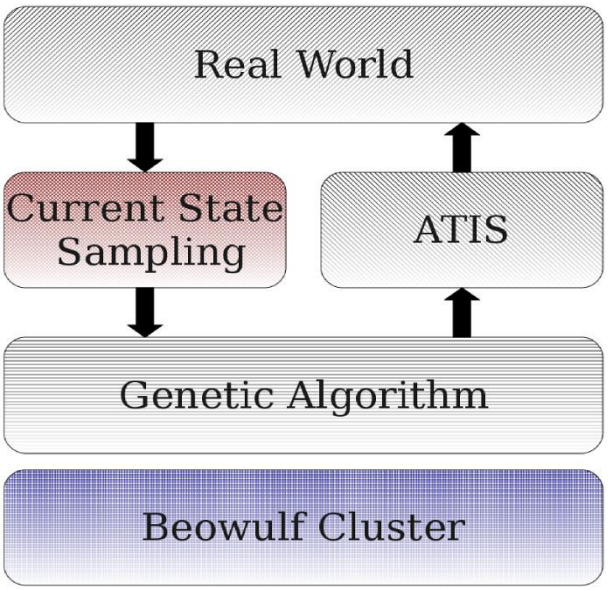


Figure 1. Overall concepts of the AETROS model for traffic route assignment

These two information sources are obtained from the "Current State Sampling" hardware/software module, which is launched every T_{period} seconds. A Moving Horizon approach using a parallel genetic algorithm searches for the optimal next cell at the crossroad cells, which are defined at every intersection, depending on every possible destination. The genetic algorithm needs a proper way to estimate if a combination, or chromosome, is better or worse; that is done with a carefully selected fitness function. In this case, for every chromosome, a microscopic simulation was launched, and a set of significant variables were sampled to associate every chromosome with a fitness value. Equation 1, the fitness function, is expressed as:

$$\mathrm{f} = \sum_{\mathrm{i}=1}^{\mathrm{N_{CR}}} \sum_{j=1}^{N_{OUT}} \frac{N_{free}(i,j)}{N_{free}(i,j) + N_{occ}(i,j) + K_a \times N_{accid}(i,j)} \Big/ N_{CR} \times N_{OUT} \tag{1}$$

where the number of free cells, occupied cells, and cells with accidents are N_{free}, N_{occu}, and N_{accid}, respectively.

The parallel genetic algorithm (PGA) runs on a Beowulf cluster, which is a multiple instruction/multiple data (MIMD) multicomputer having very good cost and performance. The resulting combination is transmitted to drivers using ATIS, for example, variable message signs, smart navigation devices, and so forth. Just by adding nodes to the Beowulf cluster to the proposed model, a wide range of traffic-network sizes can be tested without writing a new line of code. The microsimulator used for the fitness function is described in detail in Section 3, and the genetic algorithm in Section 4.

It is important to note that there is a strong real-time restriction in this architecture. The GA must finish its work in a less than T_{period} seconds to be able to give the next combination on time to the control module. That is why it is so important to have a parallel and scalable architecture. Two kinds of processes are running, a GA Master process, which performs all the standard tasks a GA should do except the evaluation of the potential solutions (chromosomes); and evaluations that are run by GA Slave processes in parallel. The parallel evaluation done by GA slave processes consist of cellular automata microsimulation in which some parameters are sampled, as is explained in Section 3.2.

The steps of the dynamic travel-route-assignment model is summarized as follows:

1. A snapshot of the current traffic situation is captured.
2. A genetic algorithm is run to ensure that the optimal combination for the 'preferred next cell' is obtained in less than T_{period} seconds. The GA steps are:
   a. A combination population is created.
   b. For each combination or individual, the future traffic situation is simulated for a number of T_{period} iterations.
   c. Using performance measurements from the predicted traffic situations for the future, every individual is assigned a fitness value.
   d. According to the fitness value, the population is ordered. Then, the GA algorithms operators are applied to the population – Crossover and Mutation – in order to produce a new combination population.
   e. The genetic algorithm evolves during a number of generations.
3. An optimized 'next suggested cell' for every crossroad cell and destination is produced by the GA and sent to the traffic control module, which displays it in the ATIS.
4. The first step is repeated *ad infinitum*.

So far, this model has been tested in a simulated environment. Currently, cooperative agreements with city local governments are being explored in order to implement the model, at least partially and progressively, into a real world network.

3. Microscopic Traffic-Simulation Model

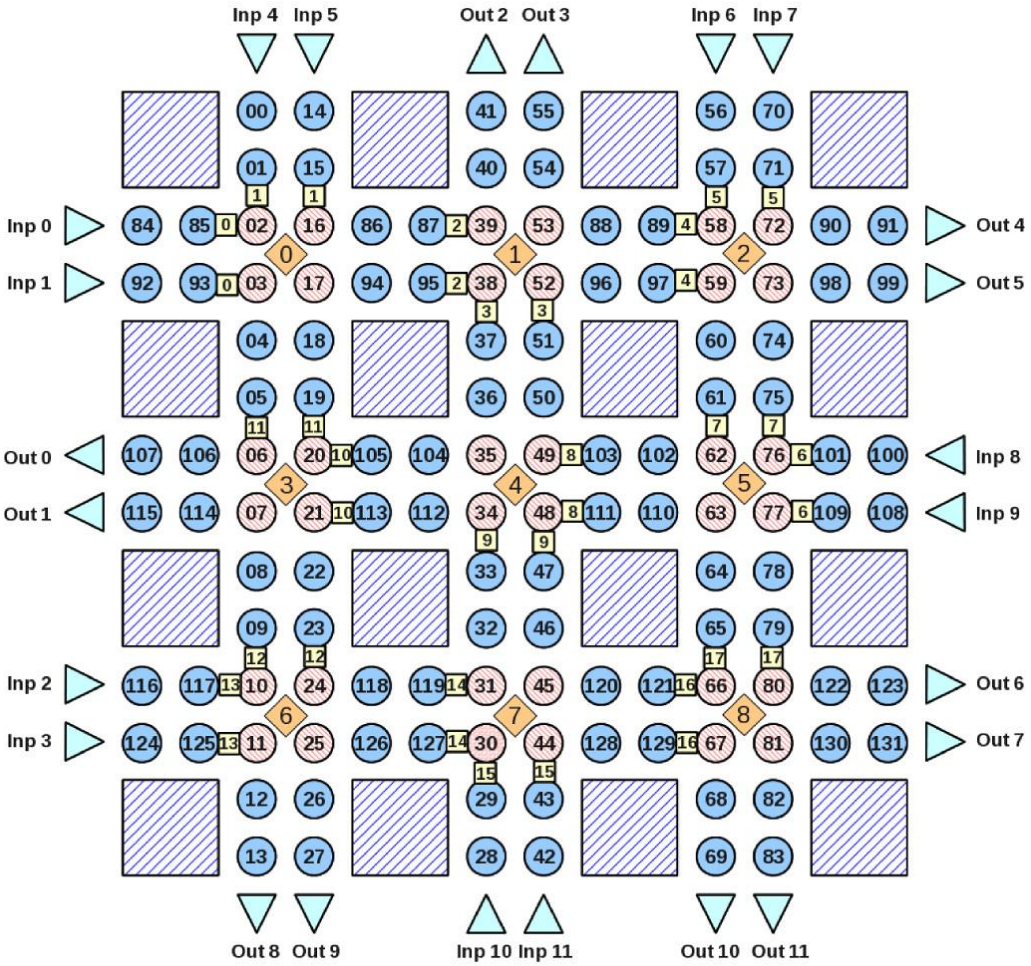


Figure 2. Discretized traffic network. Circles represent the cells, or possible positions for vehicles, triangles represent the inputs and outputs, squares indicate traffic lights, and diamonds indicate network intersections

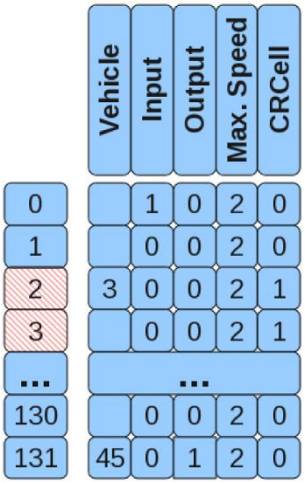


Figure 3. Data structure of the cells in the discretized traffic network shown in Figure 2

The microscopic traffic model developed in this study is a Cellular Automata model that has a set of new abstractions, making it more flexible and more comparable to the actual traffic. First, a traffic network was discretized into a set of cells, formulating a graph similar to the one shown in Figure 2. As is usually done, a cell was sampled every 7 meters. In this figure, circles represent the cells, or possible positions for vehicles. Not more than one vehicle per cell is permitted in the model. Triangles represent the inputs and outputs. The inputs are places where the new vehicles are created. Section 3.2.4 explains the vehicle creation times and the parameters assigned to every new vehicle. Squares indicate traffic lights, which so far, may be in one of two states: red and green. Section 3.2.3 describes the control policy applied to traffic lights. Diamonds indicate network intersections. Every traffic light is assigned to an intersection.

The main feature of the proposed model is how the vehicles move. It is assumed that an ATIS was available for the network. Vehicles at every cell has a set of possible next cells to which they can move, depending on their destinations (output). There are two kind of cells: 1) plain cells, for which the next possible cells are in the same set, no matter what is the destination of the vehicle; and 2) crossroad cells (marked with a pattern), for which the set of possible next cells may vary, depending on the destination of the vehicle. Section 3.2.1 explains in more detail how vehicles move in the simulation.

In addition, time is discretized in the microscopic simulation. In this case, a time step was assigned to a second, where 1 ≡ 1s. This is the sampling period that is conventionally chosen because it allows greater ease during calculations and it represents the reaction time of a human driver, more or less.

3.1. Data Structures

In this section, the main data structures used in the proposed model are described. Figure 2 we represents an example of a discretized traffic network. In the model, every cell was treated as a row in the matrix, represented in Figure 3. For every cell, a vehicle present at that cell was stored, whether an input or an output as well as the maximum allowed speed at that cell and if it was a crossroad cell, represented by being filled with hatching in the figures.

Figure 4 represents the data structure used for the vehicles currently in the network as well as for those that already left the network. For every vehicle, its position (current cell) is stored, its destination number (output), its speed, the time step when it was introduced into the network, and the time step when it left the network, in case it already left.

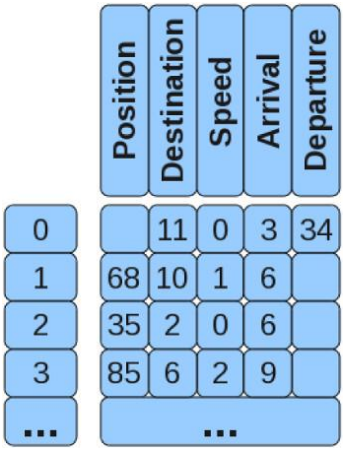


Figure 4. Data structure of the vehicles in the discretized traffic network shown in Figure 2

The most important data structure in the proposed model involves the paths. In this abstraction, paths are stored for every cell in the network, and for every possible destination, the allowed next cells. If the cell is a plain cell, a path is useful to simulate overtaking another car. When a vehicle cannot go on to the cell ahead, it may try alternative cells in the set of possible next cells. If the cell is a crossroad cell, the same rule applies. However, in order to reach the same destination, a vehicle may choose among a set of different routes. The adaptive-control model proposed in this study recommends an option by means of a convenient ATIS. If the suggested cell is free, then the vehicle will continue to the proposed cell. This abstraction is represented in Figure 5, which includes all the possible next cells for every cell. This figure also represents all the path information for the traffic network.

In Figure 6, the data structure for the paths is partially represented, and may contain that information.

Other data structures are used by this microscopic simulation model. The more important ones are as follows.

Updating_Order, **Dependencies**, **Dependents**, **and N_Dependents**. Updating_Order simply is the order for updating the vehicles in the network. A recursive routine was developed to calculate that order, intended to avoid possible problems. This data structure is described in detail in Section 3.2.2. Dependencies, Dependents, and N_Dependents are auxiliary data structures to the calculation of Updating_Order.

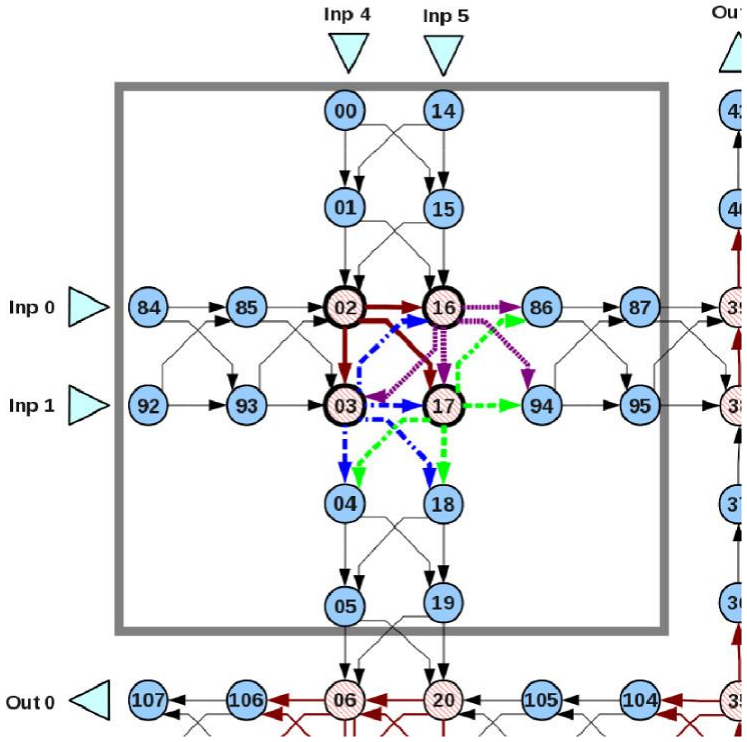


Figure 5. Paths in the improved Cellular Automata Model for Intersection #0 of the traffic network shown in Figure 2

00	0: 1 15	1: 1 15	2: 1 15	... 11: 1 15
01	0: 2 16	1: 2 16	2: 2 16	... 11: 2 16
02	0: 3 16 17	1: 3 16 17	2: 3 16 17	... 11: 3 16 17
03	0: 4 16 17 18	1: 4 16 17 18	2: 4 16 17 18	... 11: 4 16 17 18
04	0: 5 19	1: 5 19	2: 5 19	... 11: 5 19

Figure 6. Data structure for paths of the traffic network shown in Figure 2

Input_ProbMatrix and Dest_ProbMatrix. Two types of microscopic simulation were developed, deterministic and stochastic. Both types need to know when to create new vehicles and which destination will be assigned to every newly created vehicle. Input_ProbMatrix includes the probability of creating a new vehicle during every simulation period.1 This probability may change from time to time in order to simulate a variable demand across a lapse of time. This data structure can be used in such a way that the simulation may be fed with current input probability values from the real traffic network. Dest_ProbMatrix works in a very similar way. It contains destination probabilities for every input, and also can contain inferred values from current traffic in the network.

CR_Cells, **CR_N_Options, and CR_option**. This simulation needs to know which cells are crossroad cells, how many options are available for every crossroad cell, and the current preferred option. The preferred option is calculated by means of genetic algorithm optimization.

Cells_TLs, **TL_Queue_cells,** and **Inter_TLs**. In this adaptive traffic-control model, traffic is controlled mainly using a combination of preferred option for all the crossroad cells. In addition, traffic lights at intersections need to be included because of safety reasons as well as to avoid the Queue Spit Back Blockage (QSBB) effect. Section 3.2.3 describes this in detail. In these data structures, which cells may be affected by traffic lights have been stored as well as which traffic lights are related to every intersection and those cells making up the queue of every traffic light in the network.

3.2. Simulation

The microscopic simulation model developed in this study works by updating of a variable set of vehicles running over a graph of possible positions or cells. The microscopic traffic simulator is an abstraction of reality based on a great many simplifications. In real traffic, when two cars try to occupy the same position at the same time, human drivers find a way to decide who will go first. However, virtual drivers are much less intelligent, so eventually, intersections may get blocked, provoking a chain reaction and finally collapsing the whole network. This effect, known as *queues spillback blockage*, a problem during traffic simulation, and can handled by two means, the updating order (Section 3.2.2) and the policy for controlling traffic lights (Section 3.2.3).

1 A period consists of a set of simulation time steps. For every period, the next optimized combination for the traffic network is calculated.

A crucial part of the simulation model is the creation routine for new vehicles (Section 3.2.4). The control model is supposed to be adaptive with regards to a changing traffic situation. Traffic is an intrinsic stochastic process, the simulation model needed to reflect a consistent re-creation of every periodic traffic snapshot. To do so, two issues had to be addressed. First, we need to know every time we want to all the vehicles in the network, their position, speed and destination. This was a really difficult task. Second, for every input to the network, the current origin-destination probability matrix needed to be known as well as the current arrival frequency and the mean speed.

3.2.1. Vehicles Updating

Every time step all the cells in the traffic network were visited, and all the vehicles in that cells were updated. The cells were inspected in a carefully chosen order, called the *updating order*. Basically, the update order was designed to reduce potential deadlocks and queue spillback blockage. How the updating order was calculated is explained in Section 3.2.2.

The vehicle updating process included two phases, 1) updating its position and 2) updating its speed. Depending on the current speed of the vehicle, its next position was calculated. For instance, if its speed was two cells per time step (or per second, in our case), the vehicle was positioned two cells away. To do so, paths data structure was used to determined which were the next possible positions/cells in that vehicle's destination route, as has been shown in Figures 5 and 6. If the vehicle could not find free cells when considering all the alternatives in order to reach its destination, it must stay and wait in the current cell. If there is a red traffic light ahead, it will has to stop and wait.

After the next position is calculated, then the vehicle's next speed is calculated. To do so, and to avoid abrupt speed changes, the future free cells are considered. If there are free cells ahead the vehicle, then the speed is allowed to increase up to a maximum legal value. If there are not free cells for future movements, or there is a red traffic light ahead, speed is decreased or set to zero.

3.2.2. Updating Order Calculation

Using the paths data structure, for a single cell, one may know how many neighboring cells depend on it to move on. In other words, how many cells have among their next possible cells the observed cell? For this model, recursive routing was designed to explore the entire graph by following neighboring connections and determining the number of cells that could depend on a single cell in order to move on in future. Figure 7 depicts a number of dependent cells for Intersection #0 based on the paths as displayed in Figure 5. Note that every cell may have a number of dependent cells even higher than the total number of cells in the network. This is because the recursive routine may visit the same cell more than once following a different path. Common sense indicates that it would be wise to update first vehicles in cells with a higher number of dependent cells.

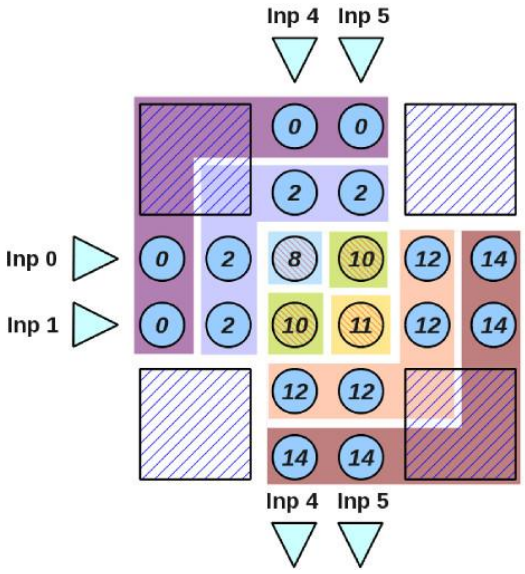


Figure 7. Number of dependent cells for every cell for Intersection #0

However, as stated in Section 3.2, the queue spillback blockage effect needs to be taken into consideration. Therefore, another criterion was added for the updating-order calculation. For a fixed number of dependent cells, crossroad cells are updated first.

3.2.3. Updating the Traffic Lights State

To prevent queue spillback blockage, both in the simulation and in real traffic situations, this adaptive traffic-control model includes a local optimization policy for traffic lights that consists of opening the traffic lights for the most crowded queues. In the simulation, for every intersection are a set of traffic lights, and for each traffic light, a group of cells were identified as part of its queue. These cells are positions at which point there is no other option but to pass through that traffic light.

As mentioned in Section 2, this model divided time into periods. In every period, the current traffic situation is passed to the genetic algorithm, which replies with the optimized combination of preferred crossroad cell options to be applied in the next period. In every simulation period, the number of vehicles waiting in every traffic light queue is sampled, and the set of traffic lights having the longest queue is turned to green at every intersection. This local optimization policy attempts to reduce queues and prevent queue spillback blockage from growing out of control.

So far, only two states for traffic lights have been implemented in the simulator, red and green. In real-world applications, the control system must include transitional states – for instance, amber – when turning from green to red. However, the optimization can be assigned fixed size slices to those transitional states.

3.2.4. Creation of New Vehicles

A crucial part of the simulation model is the creation

routine for new vehicles. The control model is supposed to be adaptive regarding the changing traffic situation, and traffic is an intrinsic stochastic process. Therefore, demand is variable. For the simulation model to consistently re-create every periodic traffic snapshot, two issues needed to be addressed. First, the current vehicles in the network, their position, speed, and destination needed to be known. In this study, it was assumed that that information was provided by a system such as ATIS. In fact, this model assumed that there was an ATIS to inform drivers of preferred options at every crossroad cell in order to reach their destination. Second, the current origin-destination probability matrix, the current arrival frequency, and the mean speed at every input to the network needed to be determined.

For creating a new vehicle routine, three variables were used, 1) the current Input_ProbMatrix data structure, 2) the Dest_ProbMatrix data structure, and the average speed of that vehicle. Input_ProbMatrix provided the probability of a new vehicle arriving at every input. This variable was used to adjust the incoming flow to the network according to reality. It is well known that the demand for each network varies across by day, by week, and so forth. Moreover, it may change unexpectedly because unforeseeable factors, such as crashes and vehicle breakdowns. Using Input_ProbMatrix, when to introduce a new vehicle at an input can be calculated. However, its destination needs to be known. This information is obtained from Dest_ProbMatrix, the probability of every destination for each input is stored. This information changes with time, and the system has been designed considering that change. Finally, every newly created vehicle is assigned an average speed.

3.3. Beowulf Cluster

The architecture of this system was based on a Beowulf cluster due to its price/performance relationship and the possibility of employing open-source software. This is a very scalable MIMD computer, a very desirable feature in order to solve all sorts and scales of traffic problems. For this research project, an eight-node cluster was set up, each node consisting of an AMD Opteron64. The nodes were connected using a gigabit ethernet backbone. Every node had the same hardware except the master node, which had an extra gigabit ethernet network card for an ‘out world’' connection. Every node had installed CentOS (Kernel 2.6.9-78.0.13.ELsmp). For parallel programming, the installation of Open MPI (openmpi-1.2.9-1) was necessary as well.

4. Genetic Algorithm Optimization

4.1. Chromosome Encoding

Figure 8 depicts the chromosome encoding used in the model. This was represented in Figures 5 and 6 as the possible next cells for the crossroad Cells 2 and 3. In the genetic algorithm, the chromosomes included the preferred next cell for every crossroad cell (CRCell) for each output. ‘Preferred next cell’ means that when a vehicle is in this cell, it should choose the preferred next cell unless it is occupied. In Figure 8, for CRCell 2 and Output 0, the preferred next cell is 3. In the same row (CRCell 2), Output 1 also is Preferred Cell Number 3. However, for Outputs 2 and 11, the preferred option is Cell Number 16. So far, only two possible states have been considered for traffic signals: red (0) and green (1).

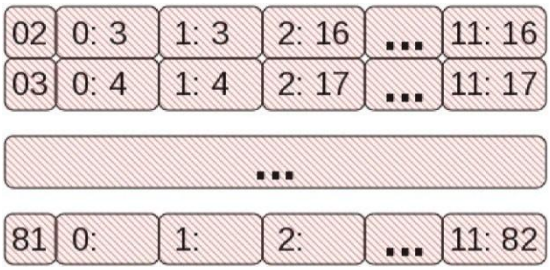


Figure 8. Example of a chromosome in the traffic network simulation

4.2. Initial Population

Every time the genetic algorithm is run, an initial population of feasible combinations or individuals has to be generated. As explained in Section 4.1, a chromosome consists of a set of preferred next cells for every crossroad cell, and for every output. In Section 3.1 the data structure CR_N_Options was introduced, in which the number or possible next positions for every crossroad cell is stored. Therefore, both for the initialization of the population and mutation, very simply, a random option is chosen within the allowed range stored in that matrix.

4.3. Random Number Generation

For random number generation, Makoto Matsumoto and Takuji Nishimura's MT19937 generator was employed, also known as the ‘Mersenne Twister’ generator, using the GNU Scientific Library (gsl-1.5-2.rhel4). This generator has passed the DIEHARD statistical tests [11]. The seeds for that algorithm were obtained from the ‘/dev/urandom’' device provided by the CentOS operating system.

4.4. Selection Strategy

A truncation and elitism combination was selected as the strategy. This meant that at every generation, a reduced group of individuals – in our case, the best two individuals – was cloned to the next generation. The remainder of the next generation was created by crossing the individuals from a best fitness subset, usually 66% of the entire population.

4.5. Crossover Operator

A standard two-point crossover operator was used. At random points – for a pair of \textit{parent} chromosomes – this selects two random points, cuts them at these positions into three pieces, and then interchanges the central chunk.

4.6. Mutation Operator

When an individual is chosen to be mutated according to

the mutation probability, the value stored at a randomly chosen position of its chromosome is overwritten using the random generator (Section 4.3). The mutation probability is not fixed. It starts with a high mutation probability that progressively decreases until reaching probability values near to the inverse of the population size at the end of the planned number of generations.

4.7. Evaluation

The genetic algorithm used for the experiments presented in this paper was equipped with a fitness function. Since this is a Moving Horizon Approach, for every real world T_{period} seconds, a slave process has to run a deterministic simulation for every chromosome, lasting for

T_{period} x $N_{\text{eva\_periods}}$ iterations, where 1 iteration ≡ 1 s. Every chromosome, consisting of a combination of next advised cells for every crossroad cell/destination pair, is used to set up a deterministic simulation at a slave process.

For every crossroad cell/destination pair, a network cells subset can be easily obtained, including those cells that can be visited in the path from that specific couple. After every simulation ends at the slave process, the resulting situation for traffic network occupancy at the corresponding cells subset is inspected. The number are counted of free cells, occupied cells, and cells with accidents or N_{free}, N_{occu}, and N_{accid}, respectively.

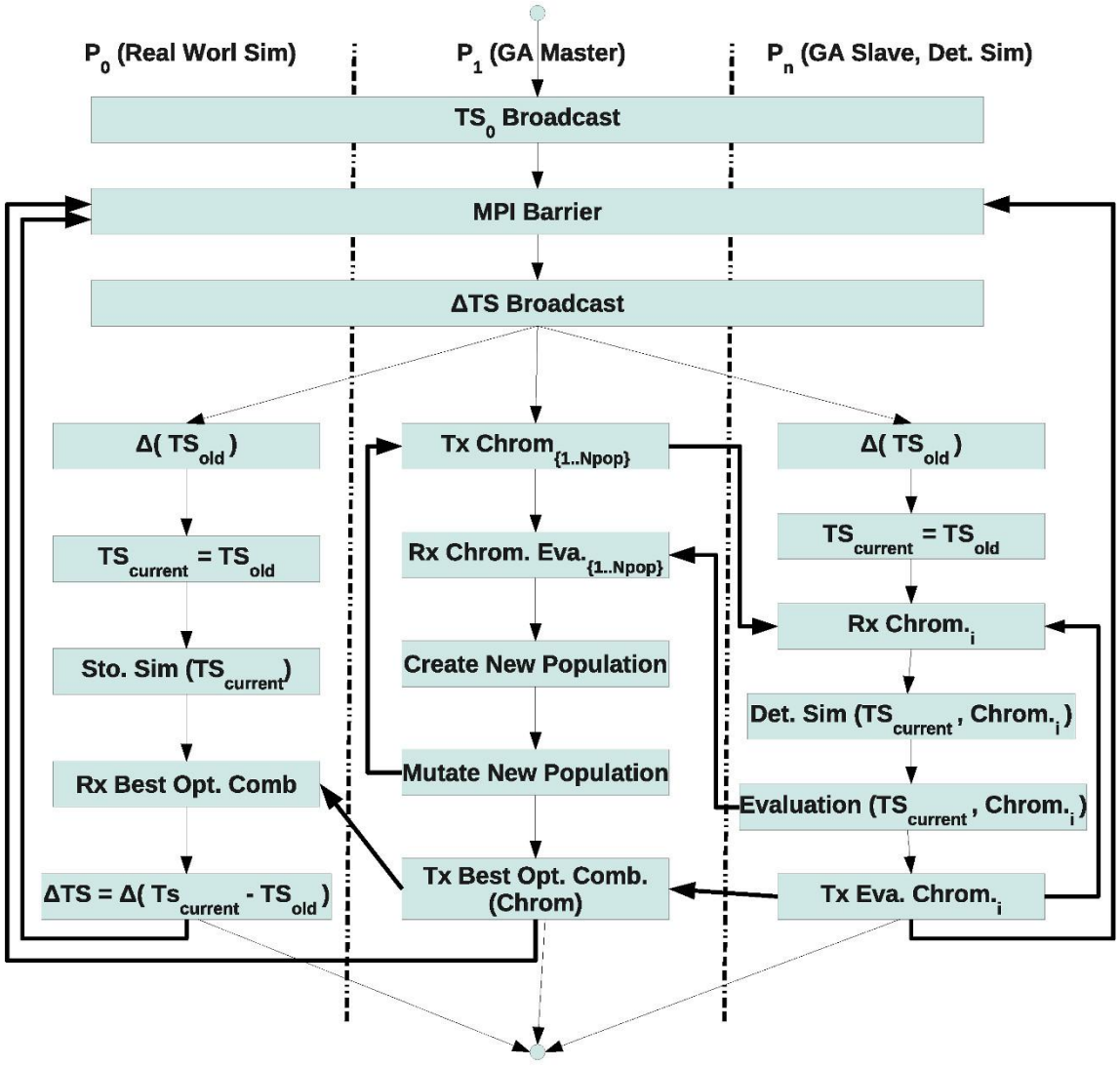
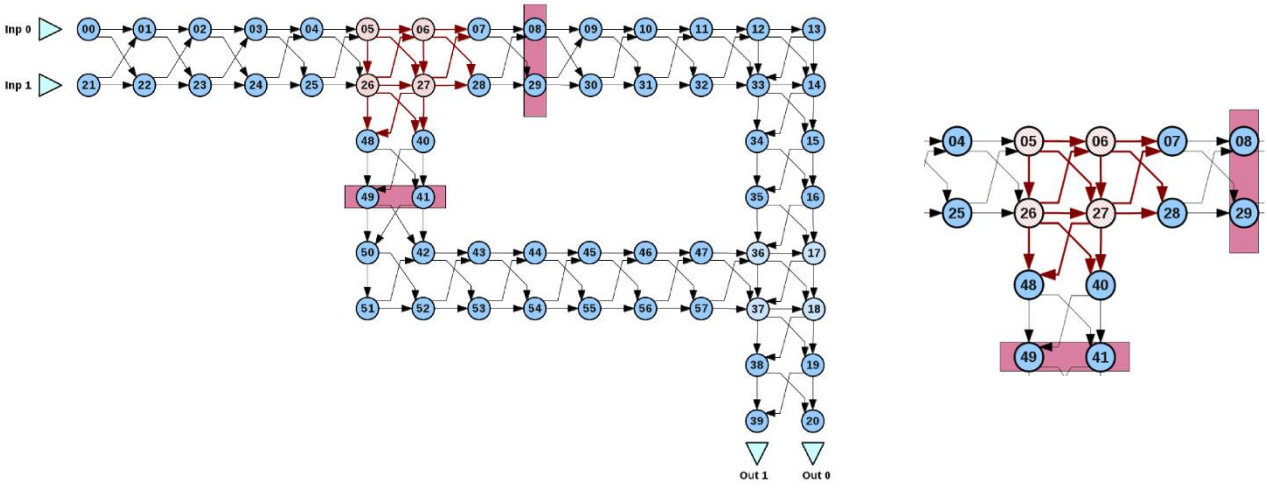


Figure 9. Parallel processes of the simulator



(a) (b)

Figure 10. Case A: A two-options network a) in its entirety and b) a detailed segment for Crossroad Cells and Forced Accident Emplacements

Equation 1 (in Section 2) depicted the fitness function used to evaluate every chromosome at the genetic algorithm that was run. This equation represented how the proportion of free cells against all cells and accidents was averaged across all the possible crossroad cells and destination couples. The number of accidents was weighted by a factor of K_a. In this study, $K_a = 10$ was used in order to give more importance to the number of accidents with respect to the number of occupied cells.

This fitness function approach was designed to be used in maximization mode. Additionally, it was designed to let the genetic algorithm to provide optimized option combinations with a higher number of free cells and/or a lower number of accidents. This was the result of experimenting with various fitness functions.

4.8. The Parallel Genetic Algorithm

The parallel program used the OpenMpi (ver. 1.2.9) library. In this application, three kinds of processes were defined, namely, the stochastic simulator, the GA master, and the GA slave processes. Figure 9 summarizes the message-passing scheme of the parallel algorithm that was designed. The duties of each process are as follows.

Stochastic Simulator - P_0 (**Sto Sim**): As will be explained in detail in Section 5.1, so far, this optimization model has not been applied to a real network. Therefore, testing purposes, a simulated 'real world' had to be developed. The P_0 process was the thread that simulated real-world stochastic traffic, the current-state sampling module and the real-time traffic-control module. In a few words, it simulates the stochastic traffic, passes the current state, and applies the optimized chromosome. To do so, it needs to communicate to the GA master process (P_1), keeping it up to date with the current state of 'real' traffic and with current traffic statistics. It needs to receive the next period's optimized combination to adaptively control traffic. It sends the start and stop signals for the entire experiment.

GA Master - P_1: The GA master process delivers the chromosomes for evaluation at the slave processes, receives the evaluations, and runs all the rest of the functions regarding the optimization, including selection, crossover, and mutation. Besides that, it receives the current traffic situation from the P_0 process and sends back the resulting best chromosome to be applied. Finally, when it receives a termination signal from P_0, it multicasts this signal to all the slave processes to make them stop.

GA Slave - P_n: The duty of slave processes – $n \in [1, N_{processes})$ – is to receive chromosomes, run deterministic microsimulations, sample the needed parameters, and send back the corresponding evaluation to the GA master, P_1.

In testing, the best speed-up[2] was obtained running a single slave process per core. There were eight dual-core computers in the Beowulf Cluster[3], as was explained in Section 3.3. Therefore, 14 free cores were available to run the slave processes.

5. Experiments

The purpose for testing the model was two-fold. First, the system needed to be checked to see if it worked, providing reasonable good options to drivers with a simple two options/two possible routes network. This was Test A, shown in Figure 10.

After that, the AETROS model needed to be tested in a quite harsh situation, using a very complex traffic network. This was Test B, shown in Figure 11. with 420 possible positions (cells), nine intersections, with no very large queueing space in the nodes links (91.43\%), 144 possible origin-destination pairs, and up to four alternative routes for

2 Speed-up occurs when a parallel program can be evaluated measuring the CPU time needed by a single process (sequential program) and by *N* processes (parallel program). The ratio between the two measures is the speed-up. Ideally, speed-up is *N* for a parallel-process program.

3 For about a 30% of the experiments, Atlante, the Canary Islands Supercomputing Infrastructure, was used due to time constraints.

every origin-destination pair. Additionally, accidents were randomly provoked, making some cells unusable for a period of time. While not extraordinary, the results from this test were encouraging, and uncovered some weaknesses in the system that will be tackled in future research.

A set of configuration parameters were fixed for both Test A and Test B. For the genetic algorithm:

- Population size (the number of individual potential combinations evaluated each generation): 200 individuals.
- Truncation surviving proportion (the subset of every GA population selected to survive attending their fitness): 2/3.
- Cloned size (number of top fitness individuals which are just cloned to the next generation): 2.
- Initial (Hyper) Mutation Probability: 0.99
- Mutation Probability factor (every generation, the mutation probability is ameliorated by this factor): 0.91

For the simulation:

- Number of evaluated T_{period} within the microscopic simulation: 4 periods
- Crash Duration Average (the average number of periods that a simulated crash lasts): 10 periods
- Crash Duration Std (the standard deviation of periods for a simulated crash): 5 periods
- Time sampling relationship: 1 it ≡ 1 s
- Space sampling distance (approx.): One cell every 7 m.

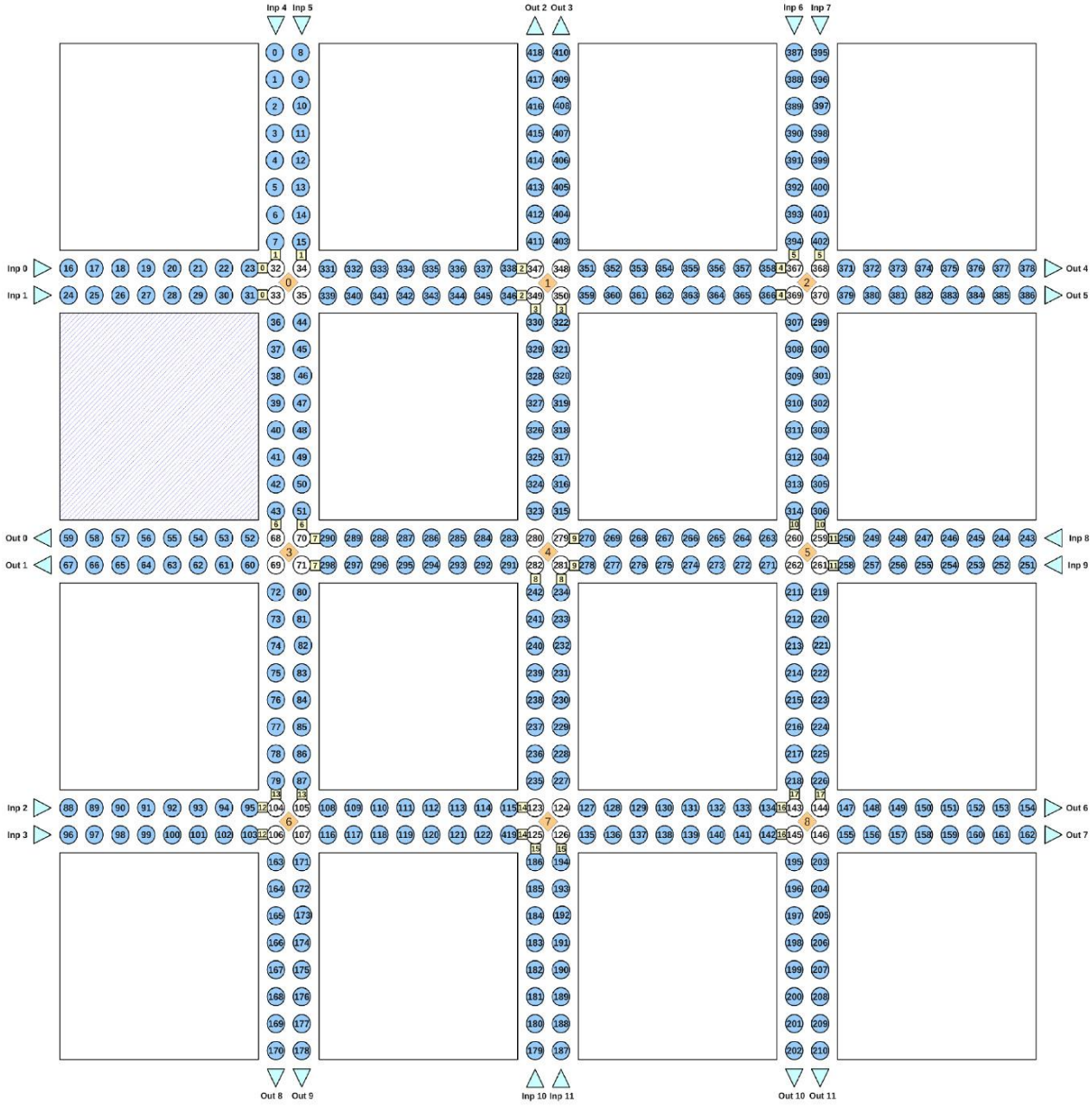


Figure 11. Test B: A large, complex network having multiple origin-destination options

Regarding the T_{period} value, 180 iterations were chosen for Test A and only 15 iterations for Test B for a faster adaptation of the AETROS system to avoid queue spillback blockage.

5.1. Test Bench Description: A Simulated 'Real' World

The adaptive traffic-control model was meant to be run over a real-world traffic network. However, it still is in the laboratory stage right now a deeply tested and safe product can be provided to traffic managers. Therefore, a test bench was required, in this case, a stochastic version of the microscopic traffic simulator.

Real traffic intrinsically is a stochastic process because of many factors. Two very important factors are human behavior and accidents. This study attempted to emulate that stochastic nature of traffic by means of some probabilistic functions. A simulated environment was used to test the proposed model. A stochastic version was performed using the microsimulation model to simulate a real-world counterpart of the system. In this simulation, using the same basis as the CA-based Microsimulator, key differences were added to give the simulated real world some realism. The following random variables were implemented:

- Random times of creating new vehicles *arriving* the network.
- A random destination assignment for every new vehicle.
- When a driver can accelerate, a random function determines whether it will accelerate or not.

For every vehicle in the network, there existed a chance to have some sort of problem that obstructs the street for a period of time. The algorithm that implemented that element used three random variables. First, a uniform variable decides when a new accident is provoked. To prefix how frequently accidents happen, a 'crash probability' value was used, a threshold value that varied across the experiments in Test B from 0.0 to 0.1. Second, if a new accident was to be set, a random uniform variable was used to choose which cell would be among the 420 of the network for Test B. Finally, the duration for an accident was decided using a Gaussian probability distribution.

5.2. Test A: The Two-Options Network

As depicted in Figure 10, Test A uses the two-option network composed of 58 cells and including two intersections, one with four crossroad cells. Each experiment in this test was divided into three phases. Phase A had no accidents in any cell during 20 T_{period}. In this case, a length of 120 time steps for every T_{period} was chosen. In Phase B, during other 20 T_{period}, accidents were provoked in Cells 8 and 29. In Phase B, 20 additional T_{period}, accidents were provoked in Cells 41 and 48. This experiment evaluated whether the suggested next option for crossroad cells reflected a system that could adapt to the provoked accidents. The fitness function was used for this test. In other words, for every crossroads cell, the number of cells occupied counted; or, with an accident, the whole graph was counted from that cell to each destination. Every cell occupied by an accident was weighted by a factor of 10 (K_a = 10). Thus, more visible effects of the accidents were expected for crossroads Cell 27, since its alternative graphs to the destination had less cells in common.

Three different probabilities for creating new vehicles were tested for each case: 0.1, 0.5, and 0.99. Table 1 shows the results for these three phases with regard to the average travel time (ATT) in time steps and the average number of vehicles that left the network (N_{VH}). These values were the result of averaging across all the results for the three input probability values and for every origin-destination pair. In this table, the first two rows show results when the AETROS system was not used. For the last two rows, the AETROS system was on, optimizing the next better options for the crossroads cells.

Table 1. Average Travel Time (ATT) and Average Total Number of Vehicles that Left the Network (N_VH) for the Three Phases of Test A

AETROS		Phase A	Phase B	Phase C
OFF	ATT	13.5882	13.6008	13.5999
OFF	N_{VH}	1411.7	1406.8	1404.5
ON	ATT	13.8557	13.9280	13.9384
ON	N_{VH}	1743.3	1728.4	1716.7

Table 2 represents the maximum, minimum, and median differences between using AETROS or not using it for the ATT and N_{VH}. In separate rows, these differences were calculated, first taking into consideration three phases and then only Phases B and C (those having accidents). The the value for the average travel time (ATT) is subtracted when using AETROS from the value obtained when not using AETROS because the AETROS system needed to provide a smaller value. For the average number of vehicles (N_{VH}), a higher number was expected when using AETROS versus not using it; this represented the result of subtracting the resulting value when not using the AETROS system to the value obtained when using it.

Table 2. Differences in Average Travel Time and Average Total Number of Vehicles that Left the Network Using and Not Using AETROS

	Phases	Max. Diff	Min. Diff.	Median Diff.
ATT	A, B, C	0.5679 (99/1/0/A)	-1.2581 (99/1/1/B)	-0.2877 (10/1/0/A)
ATT	B, C	0.3621 (50/1/0/C)	-1.2581 (99/1/1/C)	-0.3767 (50/0/1/C)
N_{VH}	A, B, C	1726 (10/1/0/A)	-887 (99/0/0/A)	89 (50/0/1/A)
N_{VH}	B, C	1694 (10/1/0/B)	-885 (99/0/0/B)	83 (50/0/1/C)

In Table 2, each value has the values for four elements associated with it, which are the experiment setup values. The first value is the probability value for new vehicle creation, for which 10%, 50% and 99% values were tested. The second and third values represent the origin-destination pair for which the corresponding value was calculated. The

fourth value represents the phase (A, B or C) that caused that average value.

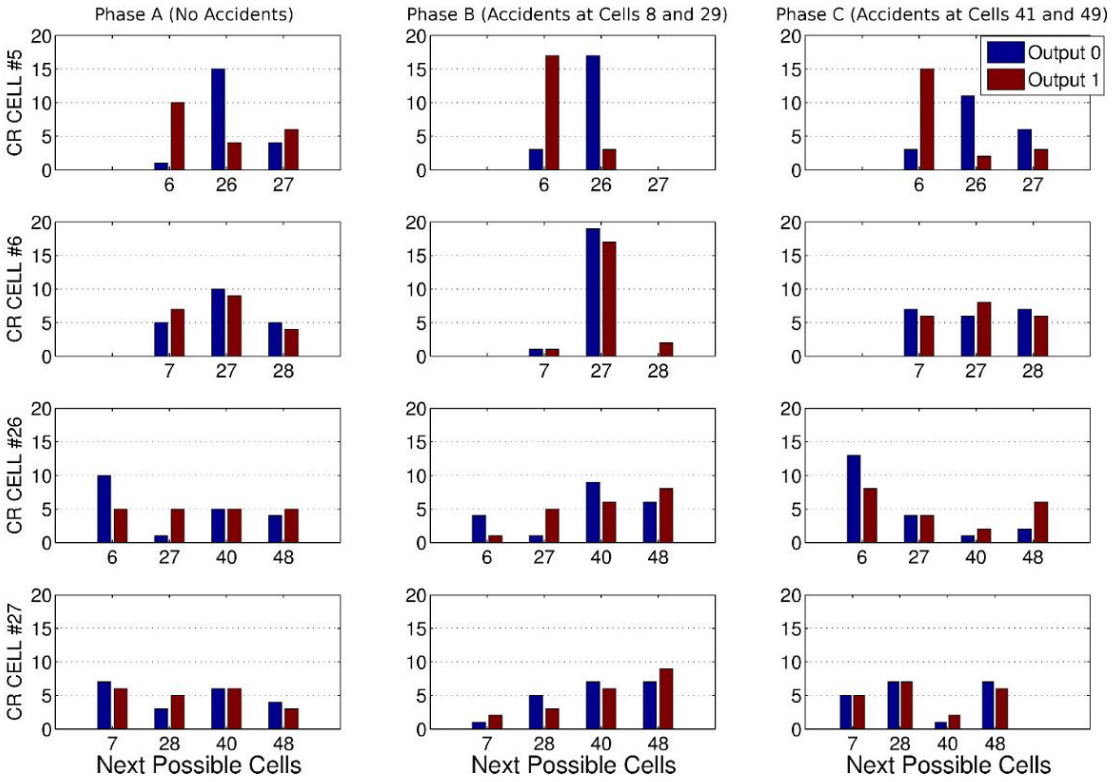


Figure 12. Input Probability 0.1

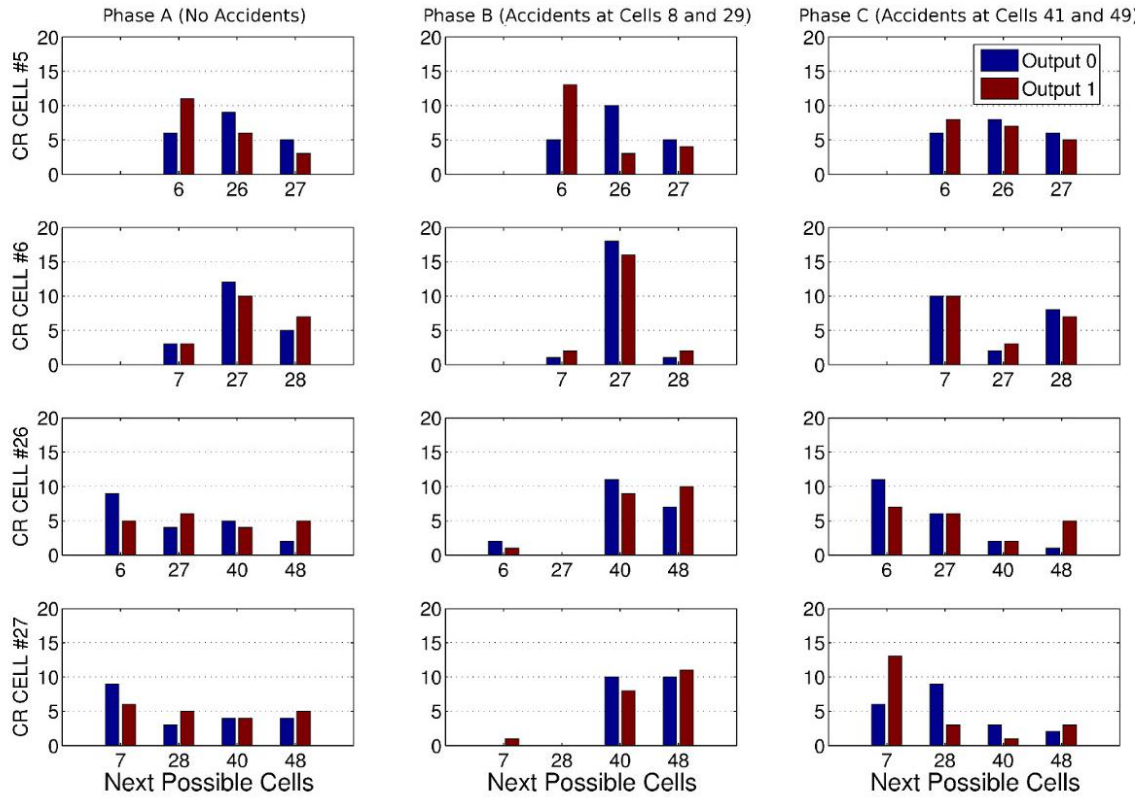


Figure 13. Input Probability 0.5

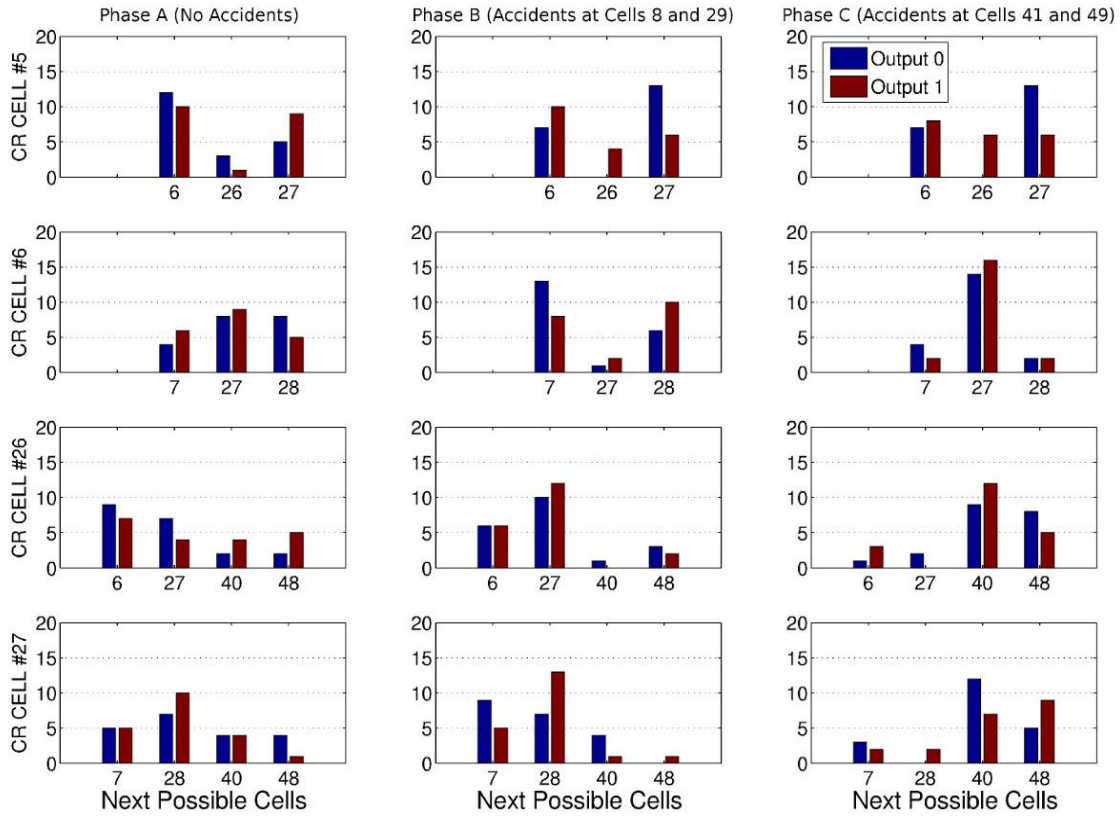


Figure 14. Input Probability 0.99

To end this set of test results with Network case A, Figures 12, 13, and 13 depcit three different demand-probability values (0.1, 0.5 and 0.99) and the corresponding histograms for the Next Suggested option produced by the GA, for crossroad cell, and the output. Figure 12 shows in detail the next possible options for each crossroad cell. For every T_{period} iteration, the AETROS system suggests one of the available options for each destination. Figure 13 and 14 show how the AETROS system naturally tends to suggest to drivers how to avoid the accidents. In those pictures, each row represents the frequency of the next cell suggestions for every one of the four crossroads cells (5, 6, 26 and 27). Regarding the columns, each column represents the resulting histograms for every one of the three phases of the experiments (A, B and C). Finally, in every histogram of these three figures, each mark has two columns, one for every possible destination.

5.3. Test Case B: A Large, Complex Network

This section presents the results of experiments for Test B, with a large, complex network. Among all the experiments performed in Test B, some provoked the network to collapse. This meant that the queues grew long enough to interrupt previous intersections, causing the queue spillback blockage effect. In a short amount of time, all the network was affected, and no car could leave the network anymore. In real traffic, drivers are skilled enough to solve this situation by somehow bending certain traffic rules; however, simulated traffic is much weaker.

For Test B, 45 experiments weren carried out, using three different probability values (input probability) for new vehicle creation and five probabilities for new accident creation (crash probability). Additionally, these values were appllied to experiments with 1) the AETROS system disconnected (labelled as 'No AETROS'); 2) with the AETROS system running, but forcing driver to choose the optimized 'suggested next cell' (labelled as 'AETROS One'); and 3) labelled as 'AETROS All', with the AETROS on, suggesting to drivers a better next option and giving them freedom to choose that cell. If the suggested option corresponded to an occupied cell, they would choose an alternative option for the same destination, in case it is free.

$$M(i,c,o,d) = \frac{N_C(o,d)\times N_{VH}(i,c,o,d)}{ATT(i,c,o,d)} \times \left(\frac{cells \times vehicles}{iterations}\right) \quad (2)$$

where *i* represents an input probability value, *c* represents a crash probability value and *(o,c)* represents an origin-destination pair Table 3 lists the experiment setups in which the network collapsed.

Regarding the Average Travel Time, Table 4 represents the mean differences resulting when using and not using the AETROS system. For every input probability and crash probability value, using both versions of the AETROS system in both versions the ATT is subtracted from the ATT obtained when the AETROS system was off, depending on if the 'suggested next option' was forced in the case of AETROS One or it was actually suggested.

In Table 5, which is very similar to Table 4, the results are presented with two exceptions. The parameter now

represented is the mean total number of vehicles that left the network for each case, and members of the subtraction are now interchanged, subtracting the values obtained without AETROS to values obtained with the two flavors of AETROS running.

The main two parameters measured in these experiments were the average travel time (ATT) and the average total number of vehicles (N_{VH}) that were able to leave the network. Considering that for Test A, the travel route trajectories are quite diverse in length, a very simple metric was devised that combined both ATT and N_{VH}, weighting each pair of values with the minimum length of every origin-destination pair. In Equation 2, this is represented by the *M* metric. In that equation, *i* represents an input probability value, *c* represents a crash probability value and *(o,c)* represents an origin-destination pair. For each *(i, c, o, d)* combination, an experiment was carried out 1) without using AETROS, 2) with the AETROS turned on, 3) allowing drivers to choose the 'suggested next cell'' or not if that suggested cell is not free (AETROS ALL), and 4) forcing drivers to choose the suggested option (AETROS One).

In Equation 2, $N_C(o,d)$ represents the minimum number of cells to be transited from origin *o* towards destination *d*. Hence, with this metric, the average travel time is somehow weighted with the *Nc(o,d)* value. In Table 4, for every experiment setup, this value has been calculated and showed the average value resulting to the subtraction of the *M* metric value when the AETROS system is turned off to the resulting M value when using the two tested AETROS flavors. Notice that for two different setups, no N_{VH} was obtained because for those two setups, no vehicles were able to leave the traffic network.

Table 3. Combinations of Parameters that Caused the Network to Collapse

	No AETROS			**AETROS One**			**AETROS All**		
Input Prob.	**0.07**	**0.1**	**0.2**	**0.07**	**0.1**	**0.2**	**0.07**	**0.1**	**0.2**
Crash Prob.									
0.0	OK	OK	OK	OK	OK	OK	OK	OK	OK
0.01	OK	OK	OK	OK	OK	X	OK	OK	X
0.02	OK	OK	OK	OK	X[4]	X	OK	OK	X
0.05	OK	OK	OK	OK	OK	X	OK	OK	X
0.1	OK	OK	X	OK	X[4]	X	OK	OK	X

Table 4. Mean Differences for Average Travel Times (ATT) with and without AETROS (iterations)

	AETROS One			**AETROS All**		
Inp. Prob.	**0.07**	**0.1**	**0.2**	**0.07**	**0.1**	**0.2**
Crash Prob.						
0.0	-4.0752	-4.8635	NaN[5]	-0.0454	0.0402	0.0391
0.01	-4.0906	-5.5904	-11.2403	0.1219	-0.1653	-0.9017
0.02	-5.7970	-7.2758	-8.0891	-0.2149	-0.5500	-1.1353
0.05	-5.5158	-9.1348	NaN[5]	0.1411	-0.0930	-0.4503
0.1	-11.7575	-12.8469	-1.5526	-0.9131	-0.1109	1.8463

Table 5. Mean Differences Regarding Total Number of Vehicles with/without AETROS (Vehicles)

	AETROS One			**AETROS All**		
Input Prob.	**0.07**	**0.1**	**0.2**	**0.07**	**0.1**	**0.2**
Crash Prob.						
0.0	104.5764	149.5417	-145.8264	1.1736	-0.4028	-0.2708
0.01	52.9028	46.5000	-139.9653	1.1319	0.2639	-38.6736
0.02	52.7708	-14.0069	-141.4861	0.8264	0.2431	-44.3958
0.05	51.4375	15.3472	-148.1667	0.3403	-0.2153	-24.4514
0.1	28.4236	-3.8403	-11.7361	-0.9167	0.7292	-10.3333

4 For some cases, though in the end the traffic network collapsed, it was all the time in the verge of not doing so.

5 For two cases when using the only one Available Option flavor of AETROS there were origin/destination pairs with not any vehicle able to leave the traffic Network. In that cases, it is not possible to properly calculate the Average Travel Times nor the corresponding M value.

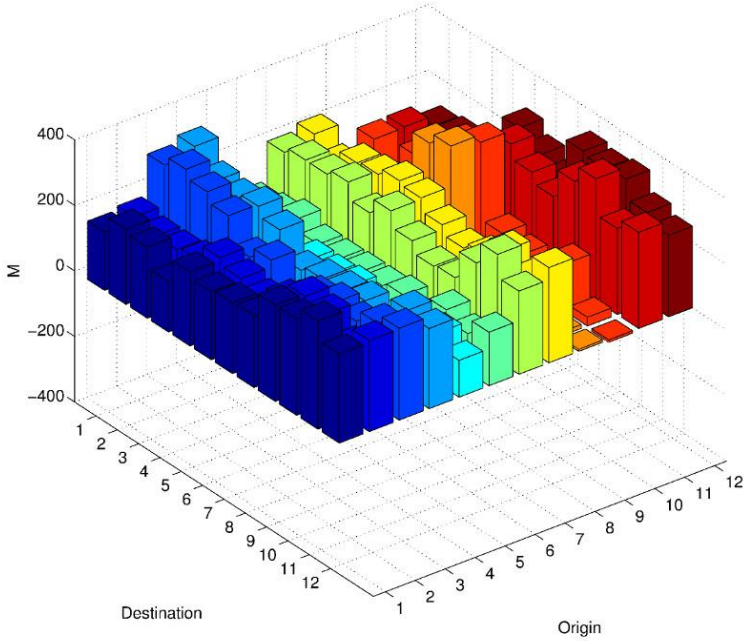


Figure 16. Best case when comparing AETROS One to results without AETROS for a 10% input probability and a 0% crash probability

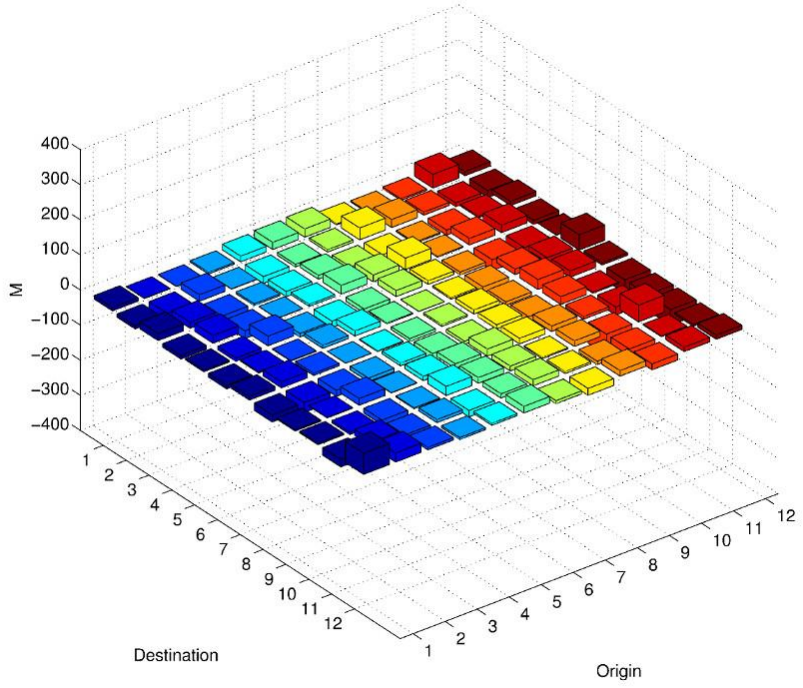


Figure 17. Best case when comparing AETROS All to the results without AETROS for a 7% crash probability

Finally, Figure 13, Figure 14, Figure 15 and Figure 16 represent all single difference *M*' values for the best and worst cases, when comparing AETROS One or AETROS All to the natural simulated traffic behavior with the AETROS turned off. As a side note, for two cases when using the only one Available Option flavor of AETROS, there were some origin/destination pairs in which no vehicle was able to leave the traffic network. In those cases, it was not possible to properly calculate the ATT or the corresponding *M* value.

6. Discussion

6.1. Test A

The experiments conducted under Test A are enlightening. They show how the AETROS system behaves in a very simple situation: two possible paths for the same destination, one with no accidents and the other with accidents in one of the two links. The evaluation function used within the genetic algorithm for every crossroad cell, the whole graph of possible next positions is explored, counting the occupied

cells, and accidents. These graph are not mutually exclusive for all the four crossroad cells, except for Crossroad Cell 27. So, better behavior of the AETROS system is expected for that cell.

In Tables 1 and 2, it seems that although the AETROS system does not improve reducing the ATT, with respect to the average number of vehicles that left the network, there clearly is increment with the proposed model. In Table 2, another clear effect is that all the worst results, displayed in the 'Min. Diff' column, seem to happen for the 99% input probability value. Therefore, that extreme input demand seems to be more than the proposed method can handle with the current parameter configurations and fitness function.

In Figure 12, Figure 13 and Figure 14, one can see how, in general, the suggested options tend to be balanced for Phase A (no accidents), and they tend to lead to paths without accidents in Phases B and C. In Figure 11, one can see the possible next options for every Crossroad cell. For instance, a vehicle in Cell 5 can move ahead towards its destination by Cells 6, 26, and 27. It does not make a big difference which next cell is chosen since from Cells 6, 26, and 27, a vehicle still can chose a link or the other. However, if one pays attention to results obtained for Crossroads Cell 27, it is more obvious that the suggested next cell is more influenced by the accidents in Cells 8 and 29 (Phase B) and Cells 41 and 49 (Phase C). This is especially obvious when the input probability[6] is 0.5 and 0.99 (Figure 13 and Figure 14).

6.2. Test Case B

From Table 3, one can easily observe that the AETROS system tends to collapse the network when the input probability demand and/or the crash probability reaches a certain level. When the AETROS system is turned off, the only traffic control running is very simple, as explained in Section 3.2.3. Every intersection has two traffic lights that open alternatively. Depending on the number of vehicles in the queue before the intersection, at the most crowded queue, the traffic lights open.

In analyzing Table 3, it seems that when we increase the pressure of demand, the local traffic control can handle that level of demand; however, AETROS ends with collapsing the network when the demand pressure reaches higher levels. Why this happens needs further examination. It is possible that the global optimization developed – in particular, the fitness function used – can be fine-tuned to stay stable in harder setups. Several fitness functions have been tested; however, this approach has revealed itself as the quickest. Further, the system needs to be quick because it is designed to work online.

Tables 4 and 5 compare using and not using AETROS for the two performance variables, *ATT* N_{VH}. In Table 4, one can observe that results are slightly better when not using AETROS, especially compared with the AETROS One' version, where drivers are forced to choose the suggested option. However, this is not a big difference, considering that the worst statistic value was 12.8469 iterations and considering that Test B was 420 cells in size. However, looking at N_{VH}, the opposite situation is evident, with better results using AETROS One than with AETROS All. Therefore, the AETROS system does not seem to offer a very big improvement in the average number of vehicles that manage to leave the network.

Aiming to balance the two measured variables and to weight the ATT value by taking into consideration the paths lengths, the *M*' parameter was designed. In comparing *M*' values, much better results occurred quite consistently when using the AETROS One instead of AETROS All. Additionally, a clear result was consistently observed that 0.2 – averaging a new vehicle at every input cell every 5 iterations/seconds – is a too harsh input probability value for AETROS for this complex traffic network. We plan to focus in studding alternative fitness functions to better cope with hard demand levels. Using 0.02 as crash probability (an average of one accident every 50 iterations/seconds) seems too much to ask of this traffic network using the current fitness function.

The two former boundaries in Figure 15 and Figure 16 are the worst experimental results obtained for Input Probability 0.2 and Crash Probability 0.02. In Figure 13 and Figure 14, one can observe much better results when forcing drivers to choose the optimized option (AETROS One) instead of allowing simulated drivers to choose an alternative option when the suggested option leads to an occupied cell (AETROS All).

All in all, results are quite satisfactory when the traffic demand probability is a 7% (an average of a new vehicle every 14.3 iterations / seconds at every one of the 24 input cells of the network) when AETROS One is used, encouraging this research team to keep on pushing with the development of the presented travel route optimization model.

7. Conclusions and Future Research

Throughout this paper, some early research results have been presented about the AETROS model for traffic-network simulation. It is a new optimization model for dynamic real-time travel-path assignment, and implements a hybrid approach between two paradigms, a system-optimum and user-optimum approach. It implements and mixes Predictive Time Delay and Reactive Time Delay models.

We have presented the overall adaptive model, the microscopic simulation model and the genetic algorithm used. Additionally, we have developed some tests over a simulated environment to show the feasibility of the system. Preliminary results are encouraging.

We have developed some tests using a time-dependent stochastic user demand, and it seems the system behaves well. Under a level of demand the genetic algorithm provides preferred option combinations that make the system quickly adapt to that variations.

6 New vehicle creation probability, or in other words, the demand pressure.

We have developed two sets of experiments with it, with two simulated traffic network and two different targets:

- To probe it consistently suggests drivers not to choose a path where accidents has been detected (Test Case A, at Figure 10).
- To explore the AETROS strengths and weaknesses with a quite complex traffic network (Test Case B at Figure 12).

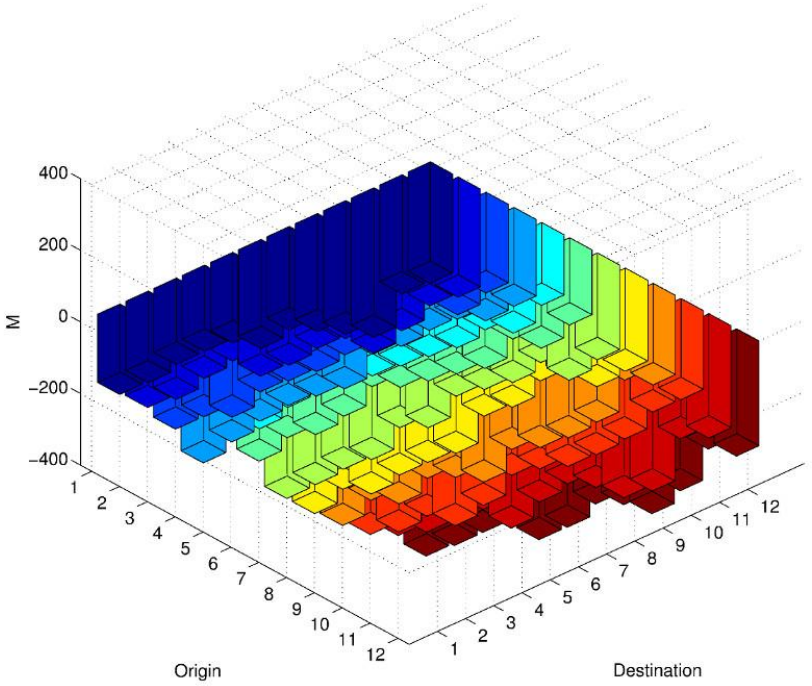


Figure 18. Worst Case When Comparing the ``AETROS One'' to Results without AETROS – Obtained for a 20% Input Probability and a 2% Crash Probability

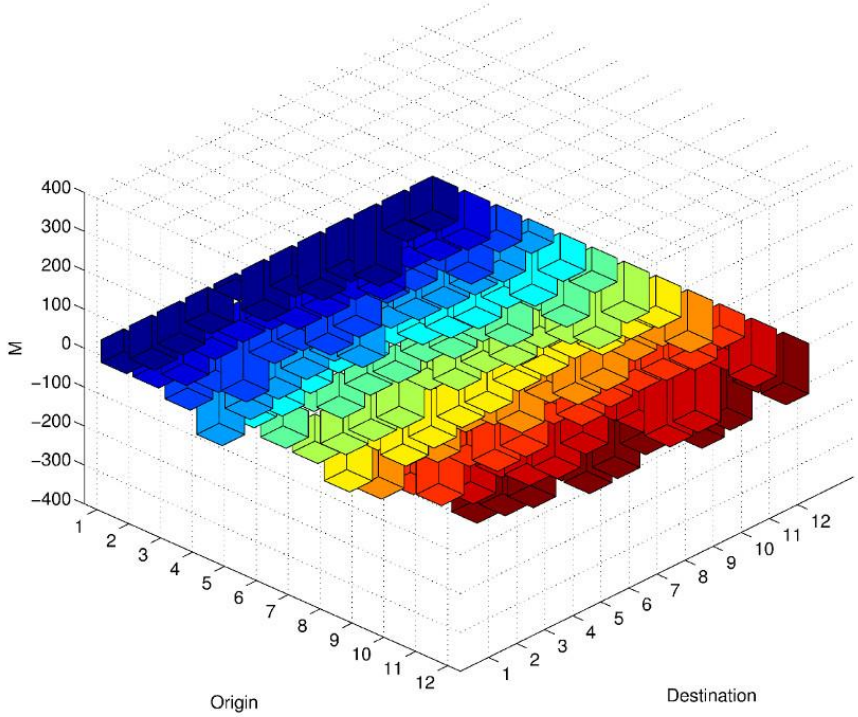


Figure 19. Worst Case When Comparing the ``AETROS All'' to Results without AETROS – Obtained for a 20% Input Probability and a 2% Crash Probability

Results are rich with information, revealing that, for Test Case B, and with the current fitness function, the higher level of traffic demand tried seems to be more than it can properly handled without overloading the network. Everything seems to indicate that the density has surpassed the Critical Density (K_C), and that is why the system is no longer stable.

With this regard we are already exploring alternative fitness function approaches in order to make the AETROS more robust in that sense. We believe that it is due to the well know Queue Spit Back Blockage (QSBB) effect, and it is meaningful that when we just use the local traffic lights control scheme, it seems to survive well with the tested demand levels.

On the other hand, we cannot forget that QSBB is only happening in our simulated real world, where simulated drivers are by far less skilled and imaginative that real human drivers. Furthermore, we are performing a global optimization, which may be maximizing an orthogonal criterion with respect a safe intersection-by-intersection local optimization criterion. The global traffic changes adaptability of the AETROS model may imply sacrificing some intersections to stay away from any QSBB scenario.

All in all, we do believe it is a path worth to be walked trying to balance both adaptability and QSBB avoidance, because for sure it is interesting to acquire the knowledge on what are the keys to tune that balance before trying it at a real world traffic network.

Summarizing, our future research plans include exploring alternative fitness functions that may make the AETROS system more robust in higher level of congestion. Additionally, we need to find a public traffic management institution anywhere for starting putting our model to the test, at least partially, with real urban traffic networks.

We plan to develop a study comparing results of including or not the crossroad cells priority as a priority in the updating order calculation.

We also plan to implement the following additional tweaks in the "Real World" simulation in near future:

- We want to implement in the rules governing the movements of vehicles the possibility for a vehicle to facilitate others maneuvers. Whether a driver facilitate or not others maneuvers may be implemented in a random variable.
- We will include amber as a permitted state for traffic lights. A random variable could rule whether a driver stop or accelerate when he/she encounters an amber (changing to red) traffic light.

ACKNOWLEDGEMENTS

The authors gratefully acknowledge the computer resources, technical expertise, and assistance provided by Atlante, Canary Islands Supercomputing Infrastructure, Red Española de Supercomputación. The Atlante infrastructure was used to develop about a 30% of the tests for this study. The authors also want to thank J.A. Longo, Technical Writer for the Howard R. Hughes College of Engineering, for editorial services regarding this paper.

IN MEMORIAM of professor Manuel Jesus Galan Moreno, who recently passed away but will stay alive in this and other works as long as his colleagues, friends and students remember his generous insightful advice.

REFERENCES

[1] Kerrigan, E. C., & Maciejowski, J. M. (2002). Designing model predictive controllers with prioritised constraints and objectives. In Computer Aided Control System Design, 2002. Proceedings. 2002 IEEE International Symposium on (pp. 33-38). IEEE.

[2] Sánchez-Medina, J. J., Galán-Moreno, M. J., & Rubio-Royo, E. (2010). Traffic signal optimization in "La Almozara" district in Saragossa under congestion conditions, using genetic algorithms, traffic microsimulation, and cluster computing. Intelligent Transportation Systems, IEEE Transactions on, 11(1), 132-141.

[3] Sánchez, J., Galán, M., & Rubio, E. (2008). Applying a traffic lights evolutionary optimization technique to a real case:"las ramblas" area in santa cruz de tenerife. Evolutionary Computation, IEEE Transactions on, 12(1), 25-40.

[4] Sheu, J. B. (2006). A composite traffic flow modeling approach for incident-responsive network traffic assignment. Physica A: Statistical Mechanics and its Applications, 367, 461-478.

[5] Gao, S., & Chabini, I. (2002). Policy-based stochastic dynamic traffic assignment models and algorithms. In Intelligent Transportation Systems, 2002. Proceedings. The IEEE 5th International Conference on (pp. 445-453). IEEE.

[6] Liu, R., Van Vliet, D., & Watling, D. (2006). Microsimulation models incorporating both demand and supply dynamics. Transportation Research Part A: Policy and Practice, 40(2), 125-150.

[7] Park, K., & Kim, W. (2001). A systolic parallel simulation system for dynamic traffic assignment: SPSS-DTA. Expert systems with Applications, 21(4), 217-227.

[8] Mahut, M., Florian, M., & Tremblay, N. (2002). Application of a simulation-based dynamic traffic assignment model. In Intelligent Transportation Systems, 2002. Proceedings. The IEEE 5th International Conference on (pp. 439-444). IEEE.

[9] Nakayama, S. I., Takayama, J. I., Nakai, J., & Nagao, K. (2012). Semi - dynamic traffic assignment model with mode and route choices under stochastic travel times. Journal of Advanced Transportation, 46(3), 269-281.

[10] Szeto, W. Y., Jiang, Y., & Sumalee, A. (2011). A Cell - Based Model for Multi - class Doubly Stochastic Dynamic Traffic Assignment. Computer - Aided Civil and Infrastructure Engineering, 26(8), 595-611.

[11] Matsumoto, M., & Nishimura, T. (1998). Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator. ACM Transactions on Modeling and Computer Simulation (TOMACS), 8(1), 3-30.