A decentralised car traffic control system simulation using local message propagation optimised with a genetic algorithm

Martin Kelly and Giovanna Di Marzo Serugendo

School of Computer Science and Information Systems
Birkbeck College, London
`jkell01@dcs.bbk.ac.uk, dimarzo@dcs.bbk.ac.uk`

Abstract. This paper describes a car traffic control simulation realised in a decentralised way by message propagations: congested nodes (roads intersections) send speed-up or slow-down messages to neighbouring nodes. Different types of journeys have been modelled: regular car journeys, accidents and emergency cars journeys. These journeys have different lengths and speeds, and affect the system differently. Optimal values of parameters, used during the simulations for controlling the cars, have been determined through the use of a genetic algorithm (GA). This paper reports as well a preliminary experiment on different simulations realised with parameters values derived from the GA.

1 Introduction

Car traffic control and monitoring systems are receiving much attention since several years because of huge and increasing traffic volume and traffic flows (e.g. 27 million journeys, and 32.7 billion vehicle-kilometres in total in London in 2004 [1]). This raises obvious issues related to ecology, economy and safety.

Traffic managements are already in place, they assist traffic officers to monitor and control traffic on cities or motorways by allowing visualisation of traffic, or analysis of real-time traffic information. Decisions are usually taken from control centres and propagated to users by the means of road signs (traffic lights, motorway screens).

This paper reports on an initial experiment towards an adaptive decentralised control solution based on local propagation of messages among nodes (roads intersections). Optimisation of key parameters has been realised by running a Genetic Algorithm (GA).

Section 2 describes the elements of the model, while Section 3 provides details about the control system and the simulations. Section 4 describes the GA used to derive optimal parameters. Section 5 reports the initial experiments realised so far. The way how such a simulation can be useful in an actual scenario is highlighted in Section 6. Finally, Section 7 mentions some related works.

2 Model

A simulation considers a set of 1000 car journeys travelling through a portion of a city during a certain period (e.g. from 8am to 10am). Traffic is controlled by locally propagating messages in the city. A graphical interface, representing a town and depicting cars, allows visual animation of the simulations.

2.1 Town model

The city is modelled merely as interlinked laneways representing individual "sides" of the road. Figure 1 shows a GUI that represents a map modelling the city as a series of dynamically adjusting throughways. The virtual city is modelled on 3km * 3km with an on-screen view of 600m * 600m, with 1 pixel equating to one meter. Vehicles are mapped as a point with a surrounding circle, and a line indicating the direction with the line length as a function of speed. Speeds are measured in metres per second and increase in discrete blocks rather than from a continuous curve - a true production system would differ by following a curve.

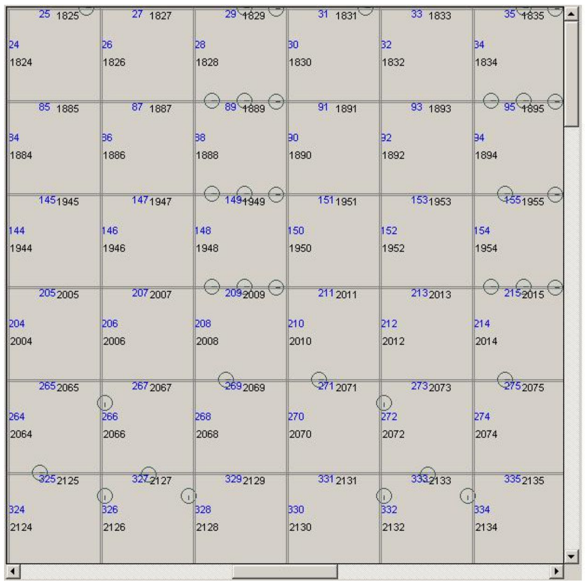

Fig. 1. Screen Shot showing a 600m * 600m portion of modelled town

Each cycle through the software model, calling each vehicle to move and consequent broadcast and evaluation of methods for up to 1,000 vehicles is regarded as a *virtual second*, i.e. one complete iteration of all city artefacts.

Journeys. A set of 1000 pre-established journeys (or routes) has been established. A journey is a set of paths which are followed from start to finish. Table 1 shows the distribution of the 1000 routes according to their length. Journeys (or routes) are of varied length and speed and may occur in parallel. They typically represent the journeys taking place in a given portion of a city during a given time slot.

It should be noted that the set of 1000 routes is auto-generated, and therefore not necessarily the greatest distribution of journeys. In an attempt to better distribute the routes within the 3km * 3km block, the generation algorithm chooses equidistant points across and down the grid as starting points for journeys. Naturally this does not eliminate favouritism with some routes more heavily travelled than others, but neither does it blandly plan routes so that all traffic is distributed perfectly, this better represents actual traffic patterns without being completely random.

Distance (km)	1.5km	1.8km	2.0km	2.2km	2.6km	2.7km	2.9km	3.1km	3.3km	3.4km
# Routes	119	122	144	131	90	88	80	74	79	73

Table 1. Routes Distribution

Roads have various innate speeds, these speeds further vary according to the relative amounts of congestion, nearby road conditions, and the current parameters set governing the simulation. Vehicles move on one side of a road, known as a path, the path dictates the velocity of member vehicles. Speeds vary from 8km/h to 96km/h.

Accidents and Emergencies. In addition to regular vehicles that go at the speed dictated by the lane, the model supports accidents and emergency vehicles: emergency vehicles have a higher speed, while offending vehicles (accident) are blocking vehicles. A simulation involves regular, emergency and offending cars.

Accident journeys are journeys along which an accident will occur randomly on one of the paths comprising the route. An emergency journey is one taken at a higher speed. 25 accidents will take place during the execution of each simulation, with 25 emergency routes also undertaken. Accidents temporarily block traffic flow on one path of a roadway; emergency routes block traffic on both paths, but allow the routing emergency vehicle to operate at the roads optimal speed. On exiting a path, both sides of the road may resume normal operations. The emergency vehicles, and also journeys where accidents occur, take place every 40 journeys, with the first accident occurring at journey 35 for accidents, at journey 40 for emergencies, and at 40 journey intervals for both thereafter (35 & 40, 75 & 80, etc.).

Accidents and emergencies are not governed directly by the control system; they are a condition of the environment. However they do impact on throughput and efficiency, for example when an accident occurs all vehicles behind the accident vehicle are stalled, this can block the path, and have a knock-on or accumulative effect in inbound paths. Likewise for emergencies, for any path an emergency vehicle is on, the other vehicles on both sides of the road are stopped completely. This is to model the effect of pulling cars into the side of the road to make way for the emergency vehicle.

An accident merely blocks a path for a given number of cycles, stifling progress on one or more paths. An emergency is effectively a high priority journey, which halts all vehicles on a road (both sides) while it traverses at an emergency speed.

2.2 Parameters

The main idea behind the parameters selections has been the willingness to enforce a *decentralised* control over the cars using message propagation: congested paths send messages to neighbouring paths (Rearward and/or Forward).

The four following parameters have been chosen:

- *Threshold*: the level at which a path reacts to congestion;
- *Range*: the distance and direction a path is willing to broadcast messages to its neighbours relative to its level of congestion;
- *Sensitivity*: the elasticity of the reaction to inbound messages pertaining to neighbouring paths;
- *Persistence*: the durability and significance of messages.

The parameters model a throttling process. Our goal is to maximise throughput in the system and therefore to correlate speed with volumes as terms dictate. The parameters selected above, Threshold, Sensitivity, Persistence, Range, control local velocity and its residual short-term affects. The Threshold indicates when the path will start issuing control messages, while the Range indicates how far to broadcast. The Sensitivity and Persistence parameters control how much a path reacts to a message and for how long the path remains influenced by the message respectively. Each parameter has six possible levels of expression.

Threshold. The Threshold parameter is the controlling factor related to throughput. It is determined by the ratio of the current Population on the considered path to total Capacity of the path. Threshold represents the sensitivity of the path to the rate of increase and represents: Population/Capacity. It will trigger velocity changes on the current path, and depending on the Range parameter, may trigger broadcast messages to all neighbours. Table 2 shows the different values for Threshold. For instance, if the Threshold is set at 1, broadcast messages to all neighbours will start being sent as soon as the number of vehicles on the related path will reach 10% of the whole capacity of the path. On the contrary, if the Threshold is at 6, the messages will be sent only when the capacity of that particular path reaches 60%.

1	2	3	4	5	6
10%	20%	30%	40%	50%	60%

Table 2. Threshold Parameters Values

Range. The Range parameter represents the range and direction of communications or broadcast targets for the messages sent among the paths. As soon as a path reaches the threshold rate specified by Threshold, it will start sending messages to its neighbouring paths. There are two types of messages used during the simulations: R (for Rearward messages) and F (for Forward messages). Rearward messages are almost always of the slow-down variety, forward messages are always indications to speed-up. It is left to each path to decay their messages and return to optimal state, the messages reflect a response to short-term prevailing conditions which tend to propagate traffic (rearward or forward) toward areas of less density as an emergent property.

The use of offset antagonistic pairs of messages was considered for this solution, i.e. send a speed-up message, and then when conditions alleviate, send a slow-down message. However the path controls its own destiny, may disregard messages entirely due to prevailing conditions (such as all-stop, or maximum speed reached), or may ignore messages.

Messages are actually propagated by *nodes* - the intersection points of the 3km * 3km grid modelling the city (see Figure 1). A node is simply a congruence of paths, nodes have inbound and outbound paths, they transmit and receive messages originating from paths, the node is also responsible for decrementing message range.

Table 3 shows the 6 possible cases of message range propagation. If Range is set at 1 (R1F0), this means that Rearward messages are sent to the immediate (connected) neighbours rearward the path (R1 stands for one hop - one node - Rearward), while no Forward messages are sent (F0). Similarly, if Range is set at 4, this means that Rearward messages are propagated for two hops rearward the path (R2 stands for two hops - two nodes - Rearward); and the same for Forward messages (F2 stands for two hops forward the path).

Sensitivity. Sensitivity represents the reaction of a node to inbound peer messages. It is not simply a one to one relationship: an arriving message from a neighbour does not immediately activate a reaction from the paths connected to

1	2	3	4	5	6
R1F0	R1F1	R0F1	R2F2	R2F1	R1F2

Table 3. Range Parameter Values

the receiving node. Indeed, the reaction is also affected by the rate of inbound messages: faster the messages coming in, greater the reaction as determined by Sensitivity. An arriving message partially fills a "capacitor", when the number of messages has arrived which completely fills the capacitor the message effect is then triggered. This equates roughly to the activation function in neural networks. It is complemented by the Persistence parameter which works to diminish any elevated state.

Table 4 below shows the different values for Sensitivity. For instance, if Sensitivity is set at 3, as soon as the rate of incoming messages reaches 30%, the node receiving the messages will actually modify the speed of the corresponding path according to the messages received (slow-down or speed-up). A high Sensitivity will cause the path to react strongly to messages, while low Sensitivity will need many more messages to accrue an influence.

1	2	3	4	5	6
10%	20%	30%	40%	50%	60%

Table 4. Sensitivity Parameter Values

Persistence. This is the decay rate for known state, i.e. a road may be unused but has altered state due to inbound peer messages. The Persistence rate is the rate of reduction in the excited state, i.e. how long it perseveres in the altered state before returning to the normal state. This will also affect the message capacitor (see Sensitivity parameter), since it will serve to reduce its charge. This rate may be seen as similar to the evaporation rate used for modelling ant trails.

The table below shows the different levels for the Persistence parameter. For instance, if the Persistence parameter is set at 4, the simulation will keep memory of received (slowing-down or speed-up) messages for 20s (virtual seconds) before going back to the normal state.

When messages are received and evaluated, the Sensitivity and Persistence become important. How much of a reaction a path has to a message is governed by Sensitivity, i.e. how much will it speed-up or slow-down. Persistence dictates how long a period of time that message will be remembered, similar to the pheromone trail. Persistence does not dictate the period for which messages are sent. The age of a message has significance. Older messages exert less influence than new messages in proportion to what percentage of maximum persistence the message has reached.

1	2	3	4	5	6
5s	10s	15s	20s	25s	30s

Table 5. Persistence Parameter Values

3 Control Model and Simulation

This section describes how the different simulations are done. A single simulation involves 1000 (pre-established) journeys. The parameters values will be the same for all paths in the city for a given simulation, i.e. all paths will have the same propensity to communicate, will communicate over the same distances, will persist messages for the same time, etc.

Each operational cycle represents the passage of one virtual second. Each virtual second, each vehicle moves the number of metres per second equating to the speed dictated by the path they are on.

A simulation uses the following elements:

- *Packet*: represents a vehicle;
- *Path*: represents a side of a road;
- *Route*: represents a journey (it is a collection of serially connected paths);
- *Node*: represents an intersection between paths. Nodes connect paths in a single route, and join multiple routes together. Nodes are used in the propagation of messages within the system.

Congestion Detection: from paths to nodes. Paths are sensitive to congestion according to their Threshold value. When they detect congestion accruing they raise messages according to their Range parameter. These messages are sent to the paths inbound and outbound nodes (Rearward/Forward in accordance with Range), the nodes then transmit the message to the correct paths as appropriate.

Propagation: from nodes to paths. If a node must propagate a message to the next node, this is flagged to the connecting path and the notified path contacts its outbound node (or inbound node depending on the direction of propagation of the message) with the message as necessary.

When a node receives a message (speed-up or slow-down) it decrements the messages distance attribute. It then iterates through its set of outbound paths (for a forward messages) or its set of inbound paths (for rearward messages). Messages cannot "echo" back to a node as they propagate away from the originating path in the given direction. Even if the message range was sufficiently large to allow a message to travel back to the point of origin it would not continue indefinitely as the message's distance is decremented as it passes through each node.

Reaction: change of path's speed. When the path receives the message, it adds it either to its increasing messages collection or to its decreasing messages collection based on the message type, i.e. whether it is a speed-up or a slow-down message. The path will evaluate its messages when cars enter or leave - messages are evaluated no more than once per real second to avoid thrashing unnecessarily.

When a path receives a message (to speed up or slow down) its reaction is based upon its Sensitivity parameter, this acts as a type of suppressor or multiplier (depending on the activation level) which tends either to retard message influence or to promote it.

The Sensitivity is effectively a multiplier on the stock increase or decrease in speed undertaken during message evaluation, the values range from a 10% increase to a 60% increase. It is not based on message direction, it augments the paths reaction when evaluating messages (speed-up versus slow-down, the age of the messages, etc.) to determine the current velocity for a path.

The Sensitivity rate is never "reached" per se. It is a multiplier, an augment on how the path reacts to messages. The cars themselves don't receive messages. When they move they proceed at the prevailing velocity for the current path, in a real world scenario there would be some transmission to vehicles. In the model vehicles know the set of path's which comprise their route, when they move they use the current path's velocity to determine how far to travel in metres per second.

The Persistence parameter works either to prolong or curtail the duration of message influence. Path velocity is a combination of self-determined and peer-group influenced factors. Its own velocity is a function of congestion, its peer group may influence that velocity through messaging, i.e. a road is uncongested with a velocity of 14 metres per second, however due to congestion ahead, the messages received by its peer cause the path to reduce speed by 50%, the level of persistence is similar to the rate of evaporation of ant-trails. It determines the rate at which messages' influence is reduced. If a path maintains a peer-influenced rate for a very long time it may falsely report max speed, as above, 50% lower than its true maximum. If Sensitivity is also low, an inbound path may have to raise many messages for it to alter its state. Persistence helps to ameliorate this factor over time, meaning that state will change on request (according to Sensitivity) but will decay over time and the influence of contrary messages.

Throttling messages. Forward (speed-up) messages are transmitted forward, triggered by the Threshold parameter value to attempt to clear the route forward to alleviate congestion on a given path.

Rearward (slow-down) messages are transmitted rearward to anticipate congestion, i.e. when a path fills up, the feeding path cannot enter. Therefore throttling by congestion is an emergent state, slowing down inbound paths when Threshold is hit may help to avoid a stalled path condition.

Messages serve to alter the optimal state of a pathway. The pathway has an innate tendency to maintain its optimal state, reflected in the Persistence parameter which tends to maintain or direct a path towards this state. Messages may also serve to excite or inhibit path velocity, how a path responds is controlled by the Sensitivity parameter, high values for this parameter exhibit greater response to inbound messages.

As said before, the Threshold and Range (distance) parameters are primarily responsible for the message propagation.

When a path hits a level of congestion that breaches the setting defined by threshold it initiates a message to intersecting paths, basically telling inbound paths to slow down, i.e. to feed fewer vehicles, and outbound paths to speed up, so it may feed vehicles to them faster.

The range of the message, or how far this message propagates is governed by the Range parameter.

An individual with a value of R1F2 in this parameter will propagate slow-down messages rearward to one node (and therefore to all inbound paths registered with that node), it will also send speed-up messages to its outbound node (and consequently to all outbound paths), when these paths receive the speedup message, they notice that a range counter is not zero and therefore further propagate the message to their outbound nodes. When a node receives a message for propagation it first decrements this counter to register the consumption of the message before passing it to each neighbouring path (see Figure 2).

The Persistence parameter controls the period of influence of the message, once the time is past the message is removed entirely. Also older messages exert lesser influence than newer messages (a 28 second old message is inferior to a brand new arrival as the new arrival more correctly reflects prevailing conditions). Paths use the decay rate/persistence level for two purposes: to modify path velocity appropriate to congestion levels both of itself and of the conditions of the local group; and also to control the return to optimum speed for the path. This optimum speed is the speed it attempts to maintain. When a path returns to its optimum speed through message decay, it informs any paths that it sent slowdown messages to disregard the message; thereby influencing neighbouring paths which may reflect and artificially slow velocity to snap back to equilibrium faster.

Cascading Message Propagation. When an accident blocks the path that it is on, this may have a knock-on effect on paths that connect to this path as vehicles may not be able to join as the path becomes blocked. A first node

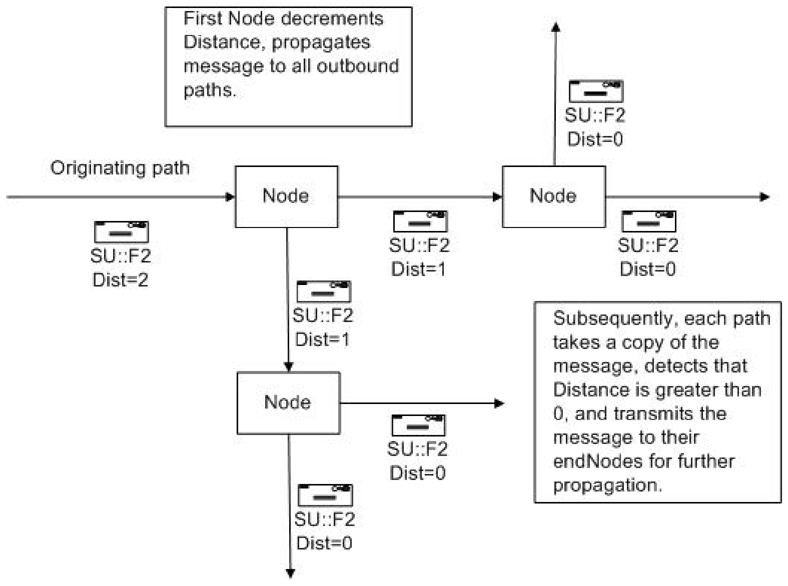
First Node decrements Distance, propagates message to all outbound paths.
Originating path
Node
Node
Node
SU::F2 Dist=2
SU::F2 Dist=1
SU::F2 Dist=1
SU::F2 Dist=0
SU::F2 Dist=0
SU::F2 Dist=0
SU::F2 Dist=0
Subsequently, each path takes a copy of the message, detects that Distance is greater than 0, and transmits the message to their endNodes for further propagation.

Fig. 2. Messages Propagation

raises a series of slow-down messages, those messages are propagated as R1 (for instance), now the receiving nodes start actually slowing down the vehicles. The nodes further up the path, or these receiving nodes themselves, will start observing congestion, and start sending messages for slowing-down, so a new series of R1 is propagated - but not from the original node.

In case of an emergency the behaviour is slightly different, both sides of a road are stopped to allow an emergency vehicle to traverse. When the emergency vehicle moves off the path to its next path, that road is freed up for travel once again, and the subsequent road is stopped, these actions continue until the emergency route is complete.

Collisions. Collision avoidance is undertaken by projecting a bounding rectangle forward from the vehicle's position; its width is set to the bounding circle indicating the vehicle's location, its length projects forward from the vehicle in proportion to the current speed. As a vehicle moves from one path to another, it ensures there is room for the vehicle at the point of entry (by calling a specific method on the target path). Otherwise as the vehicle travels it checks that it is not about to collide with a neighbour prior to executing the move. This allows vehicles to queue up behind one another. Vehicles will not take oncoming traffic into consideration when detecting potential collisions. Vehicles will overtake a blocking vehicle on the same path after a small random number of initial ”waits” this prevents vehicles who are stuck waiting to turn onto an adjoining path from blocking all vehicles behind it from carrying on forward. This represents a simple model of typical behaviour, the turning vehicle normally assumes a position on one extreme side of their lane, rearward vehicles must ameliorate their behaviour to accommodate, hence the waiting for a random number of cycles (currently set to a maximum of 5).

4 Optimisation of Parameters with a Genetic Algorithm

The purpose of genetic algorithms is to derive *solutions* to optimisation problems. A genetic algorithm starts with a *population* made of randomly chosen *individuals* each represented by its own set of *genes*. Each individual represents a possible solution to the optimisation problem.

GA Overview. Our goal is to find the set of parameters values which result in the most efficient resolution of the congestion problem. Therefore, the *optimisation problem* consists in determining parameters values (i.e. genes) that, when used to control the runs of 1000 journeys in a city, the (total) time to complete (to simulate) these 1000 journeys is the shortest possible time.

An *individual* is then a 4 tuple (i.e. 4 genes) and represents the run (one simulation) of 1000 journeys in the 3km * 3km city (see below for a description of the genes). When the 1000 journeys are being simulated the values of the 4 genes are used for running these journeys; consequently the time implied to complete these 1000 journeys is then an indicator of how "good" (fit) the corresponding individual is. The 4 gene values stand for the values of the 4 parameters of Section 2.

The *initial population* is made of 1024 (4^5) individuals. The simulations are run on those 1024 individuals. Each simulation starts with 1000 vehicles in the city (each driving one of the pre-established routes). As vehicles complete their journeys they are removed from the city. When fewer than 200 vehicles remain, the simulation stops. The fitness function is then evaluated on this simulation. The fitness function is given by the total amount of time the simulation has implied to complete the 800 first journeys.

After the initial population of 1024 simulations is evaluated, we pick the top 100 individuals, the rest is discarded. The GA crossover and mutation (2%) aspects are enabled in order to evolve the population from that point. A new generation of individuals will be produced, which will then undergo another simulation-selection-crossover/mutation process for another 1000 generations.

Each organism represents a gene sequence equating to individual genotype. The organism is tested against the same criteria, and with the same input data: the same set of 1000 journeys is used throughout the whole experiments (among which only the 800 first completed are considered for the fitness function evaluation). The testing environment only differs according to the influence of the individual organism's genotype.

The *goal* of the whole system is then to find optimal parameters values that when applied in the car control simulation allow minimising the total length (in time) of all undertaken journeys.

GA Genes. As said above, the GA genes are exactly the parameters: Threshold, Sensitivity, Range and Persistence. The possible values are those given by tables 2 to 5. Each gene has then 6 levels of expressions: this ensures a population large enough with enough variety so that evolution would demonstrate successful candidates but not so large or finely tuned as to represent a great many wasted cycles comparing with almost-like candidates. Of equal importance is not selecting a population of such small size and coarsely-grained attributes that would render evolution unnecessary for purposes of evaluation.

GA Initialisation. The GA starts with 1024 randomly chosen individuals (4-uples of genes) but with a central tendency. Indeed, i.e. the first population contains few if any 1 or 6 strength genes. This was chosen to avoid evaluation of sets of genotype with all minimum or all maximum attributes in the initial population.

GA Selection and Termination. Once the simulations (a set of 1000 journeys each) are run on the initial population of 1024 individuals (1024 different genes), the fitness function is applied on each of these simulations. It is given by the total amount of time in virtual seconds needed to complete the first 800 (out of 1000) journeys. This measurement is based on the distance travelled, and the virtual time taken.

This initial population provides a gene-pool for further searching through crossover and mutation. Indeed, a set of 100 fittest candidates is maintained for a further 1000 generations. At the end of each simulation if the current candidate is fitter than the minimum it is added to that set, the minimum candidate is dropped off the end of the list.

The process is the following:

```
select the two fittest candidates from the set of 100 individuals
loop:1000
 create two offspring and evaluate them
 take the fittest of these
 if they are fitter than the minimum, then
         add them to the set of 100
end loop
```

This requires 2000 evaluations to proceed through 1000 evolutions (1 per offspring). This also means that if the fittest candidate were in the initial population it will not be replaced.

GA Crossover/Mutation. When two candidates are identified for breeding, a random number is chosen between zero and 3, representing the point of crossover. Two new individuals are then generated from the complementary pairs due to the parent genotype split at the point chosen at random. Mutation applies to the new chromosome at a rate of 2%.

About the 800 first completed journeys. The fact that we choose the 800 first completed routes for computing the fitness function could be seen as introducing a bias factor in the evaluation of the results. Indeed, it could be argued that the worst journeys would not complete correctly, or some may be blocked somewhere in a deadlock or in a similar situation. According to the experiments done, the remaining 200 are not blocked in a deadlock as all the simulations complete for all journeys. They are cut off deliberately at this point because of the fact that on a 3km * 3km city the last couple of hundred vehicles have the streets to themselves. At this low congestion level there is no contention for resources and no comparison to be found between differing individuals. Thresholds would not be hit at such low congestion levels, and it is not a good representation of an urban traffic environment. We believe that all performance related aspects of the system will have demonstrated their fitness through the evaluation of 800 journeys, in contention for resources with up to 1000 other vehicles, and one or two journeys in the latter stages of evaluation should have no bearing on the calculation. If a set of genes is not fit, its weaknesses will have been demonstrated by mismanagement of the traffic scenario for the preceding journeys.

5 Experiments

This section describes two sets of results obtained when running the GA described in the previous section as well as a modified version of it. These results need to be complemented by additional experiments in order to further confirm and better analyse these results. However, they already show that the determination of the parameters, by running a GA, help finding optimal solutions, and thus enhancing the city performance.

Experiment 1: Fitness Function Distribution. Figure 3 shows an ordered distribution of the fitness function value over the last 256 generations (out of the 1000 initial, and 2000 (1000 * 2) generations).The fitness functions values range from 102s to 15s of virtual time. This measurement is based on the actual distance travelled, and the virtual time taken. As said above, a virtual second elapses when an entire cycle of all city artefacts has been realised (calling each vehicle to move the number of meters corresponding to one second of the speed specified by the road they are currently in, broadcast of all the necessary messages, and evaluation of corresponding methods).

The resulting data show that there are many viable (optimal) solutions, i.e. 95 of these solutions are below 30s, of which 50 are below 20s. The above figure shows an ordered distribution of the solutions, however the solutions produced by the consecutive generations do not demonstrate any convergence of the fitness values.

Experiment 2: Modified GA. A modified version of the original GA of Section 4 has been realised in order to try to observe more quickly any convergence of the solutions.

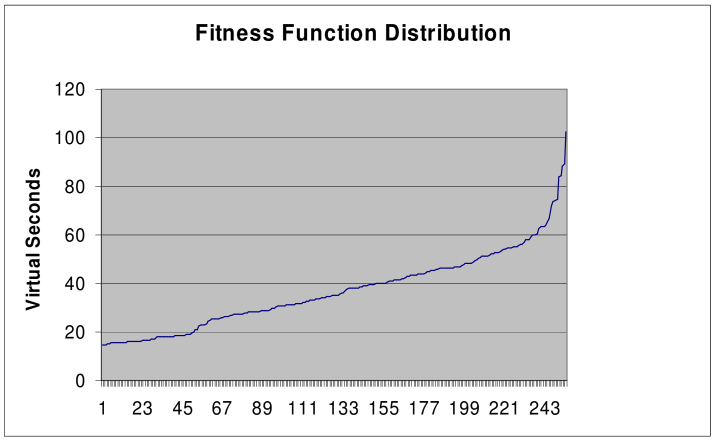

Fig. 3. Fitness Function Distribution Over Last 256 Generations

The genes types and values are the same except for the Persistence gene that now takes the values given by Table 6 below. By running the simulations, it appeared that with efficient 4-uples of genes, the fitness value function is below 20s, thus the Persistence does not need to exceed this value.

1	2	3	4	5	6
3s	6s	9s	12s	15s	18s

Table 6. New Persistence Parameter Values

In the modified GA, the first generation has been created from 100 randomly selected individuals (instead of 1024 in the original GA). A global set of the fittest individuals is maintained throughout each generation.

Next generations are produced in the following way: individuals are selected at random from the set of fittest individuals. A random number is selected to indicate the point of crossover. Each individual is then subject to a 5% potential for mutation on each gene, however only one gene may be mutated per individual. Once the set of candidates is complete, the evaluations begin again. Thirty generations were executed with values remaining stable and consistent in the later 12 generations.

The simulations of the first generation (made of 100 individuals with pure random selected gene values) show fitness functions ranging from 53s to 13s. The 12 last generations have fitness functions for the set of individuals comprised between 13s and 22s, and the last ones clearly around 17s (see Figure 4).

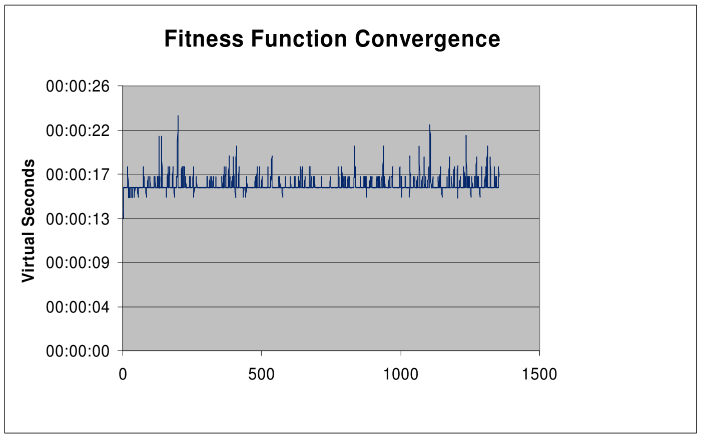

Fig. 4. Fitness Function Distribution Over Last 12 Generations

Examples taken from the best solutions obtained during this experiment, among the 12 last generations, are given by Table 7 below. High Sensitivity and Persistence are frequently observed for good solutions.

An interesting point to report is the apparently direct correlation between the Range value and the Fitness Function on *random* individuals (first generation). Indeed, Figure 5 shows on the lower part an ascending arrangement of the fitness function, and on the upper part the corresponding Range values: when the Range

	Threshold	Range	Sensitivity	Persistence	Fitness
1	30%	5	50%	18s	13s
2	30%	5	20%	9s	14s
3	40%	2	50%	15s	15s
4	60%	6	50%	15s	16s

Table 7. Examples of Efficient Genes Values

values are 3 or 5, the fitness function stays between 13s and 20s, when the Range value is 2, 4 or 6, then the fitness function raises above 25s.

However, in the last 12 generations, such a correlation does not appear (Range values are indifferently distributed among the 6 possible values). A possible explanation would be that the set of 4 parameters, modified through the GA procedure, have a combined influence on the fitness function.

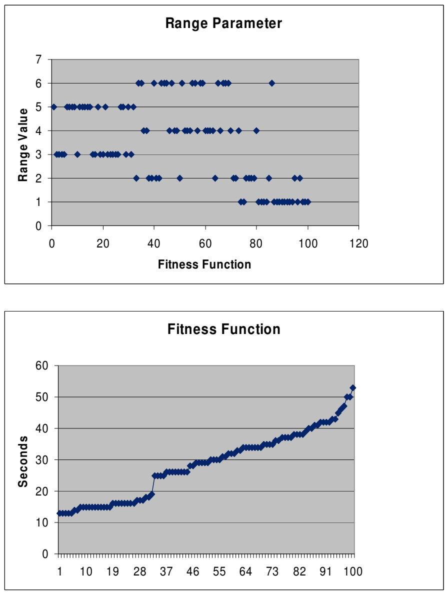

Fig. 5. Range vs Fitness Function in Random Population

A similar phenomenon has been observed on the last 256 generations of Experiment 1 as well. This lets presuppose that in Experiment 1, more cycles would be necessary before any convergence could be visible, and that the results are similar to randomly chosen individuals.

Results. Results reported here are very preliminary, and simply show (confirm) that the use of a GA is actually useful in determining specific optimal parameters further used to control the city car traffic.

Additional experiments are necessary to investigate the actual influence of the parameters, their correlation, as well as the behaviour of the city under different conditions (other sets of journeys), and under different control schema (e.g. different sets of parameters).

The current model is simple and consequently has some limitations, for instance it uses the same parameters values for the whole city portion considered. However, since the size of the considered city is rather small, this may be acceptable here. Further extensions should consider a bigger city, or a combination of smaller portions having each their own sets of genes values.

6 Towards an actual traffic simulation system

By communicating traffic levels in real-time, we could apply the decentralised message propagation system today by controlling throughput using traffic signals. In the future vehicles will be given a maximum upper or optimal speed limit, thus allowing an actual implementation closer to the proposed model. Modification to vehicles would be unnecessary: some vehicle recognition system to measure inbound and outbound vehicles would be sufficient. Such recognition systems are already available in major cities. Of course, the GA should first be run on actual (recorded) traffic values related to a well identified city zone, so that optimal parameters could be derived.

Currently, most traffic control systems are not tuned to prevailing dynamics. The system described in this paper demonstrates the improvements in efficiency capable by cooperation and coordination at local level. For example, in the system above the widest broadcast signal is two nodes forward and two nodes rearward; this avoids the complexity of centralised command and isolates the grid from massive message propagation which could flood the network. Applying the principal of decay to the message allows it to have influence on a curve rather than a simple digital switch. This is taken directly from ant-colony optimisation; it reduces the complexity of message management (i.e. having to fire messages as transactions in antagonistic pairs). It is particularly important in allowing the local conditions to adapt and recover to dynamic conditions. However it should be pointed out, that any path which sends slow-down messages to inbound nodes will always send a speed-up message to it closest inbound node once its own velocity resets to optimum.

In addition to communication among nodes, direct communication among cars can be envisaged in the future thanks to current works on wireless intervehicle communication. In this case, the system described here could be extended in different ways: to act according to the school of fish metaphor, where each car maintains a sufficient but minimal distance from neighbour cars favouring fluidity of traffic; or to propagate information along a series of cars (several hops) to support re-routing in case of congestion, etc.

7 Related Works

Car Traffic Control Systems Projects. Actual implemented systems use one or several control centres receiving real-time traffic data, and from which both manual and automated decisions can be taken and propagated to the vehicles through different means.

The technology aiming at *acquiring* real-time data related to traffic conditions is already available: traffic detectors able to detect the volume of traffic, the average speed of vehicles, time between vehicles, blocked traffic; visual control of traffic is available through video surveillance cameras. Similarly, the technology, aiming at *informing* the drivers are also quite advanced: either through specific devices on the vehicle, or through signs on the motorways.

Complete traffic managements, combining acquisition, decisions, and information to drivers are of different nature: detection of accidents, fluidity of traffic, delivery of traffic information to drivers via broadcasting media such as radio; enforcement of traffic policies (e.g. London congestion charge).

The city of Athens has a particularly interesting complete system, built for the Olympics, which uses cameras, an airship, and distributed command to control traffic light phases and durations[1]. From one hand, it incorporates data traffic acquisition using diverse techniques such as TV cameras or ground detectors; official vehicles used as mobile sensors to continuously monitor traffic recording of traffic volume via sensors. On the other hand, a decision making system designs solutions to solve traffic problems, and sends automatically corresponding information to vehicles on the road (traffic signals), or alerting police. Several similar projects are currently under way in large cities such as Beijing or Hong Kong.

Adaptive control strategies. Several works can be mentioned that make use of different techniques in order to provide car traffic control strategies that adapt to real-time situations.

Swarm-based traffic control usually employs ant metaphor for inducing a decentralised traffic control. We can cite [3] who apply the pheromone metaphor to provide a decentralised traffic congestion prediction system: cars deposit pheromone along their route which is later retrieved by forthcoming cars. The amount of pheromone deposited depends on the speed of the car, and represents the density of traffic: low speed produces high concentrations of pheromone, while high speed produces low concentrations of pheromone. The amount of

[1] http://www.roadtraffic-technology.com/projects/athens/

pheromone later retrieved by other cars provides an indication about traffic congestion and thus serves for short-time traffic predictions.

Similarly [4] uses the ant metaphor to communicate among cars and to provide a simulation of traffic dynamics in different scenarios. In this case, an additional evolutionary algorithm, including a swarm voting system for preferred traffic light timing, is introduced in order to minimise the average waiting time of vehicles.

As an alternative to swarm-based techniques, we can mention the works on intelligent transportation systems focusing on self-managing and self-organising solutions based on service-oriented architectures and the corresponding middleware[2].

As far as the use of genetic algorithm is concerned, we can mention works at the Utah Traffic Lab on the use of GA for optimising traffic signal timings[3]. This work makes use as well of a simulation used for evaluating the different identified timing plans; or the works of [2] reporting as well the use of GA for optimising light traffic timing. GA have also been used for finding optimal solutions to air traffic control, for instance [5] uses GA to derive flight plans and optimise time-route problems.

8 Conclusion

This work will be continued and the model, the GA and the simulations will be revised. A series of future directions for enhancing this work have already been identified.

Evaluations will have to extend past a mere 800 journeys, and instead may focus on time-slices with fitness evaluated on the throughput achieved during each period. We would want to measure system performance under different conditions and states, and evaluate each of these with different sets of parameters. This is achievable by adding some conditions governing routes available and vehicles available to undertake those journeys, in different geographical areas, for each testing phase.

The initial population of 4^5 individual has been chosen randomly among the total possible population of 4^6 (4 genes with 6 level of expressions). Future versions will likely introduce performance improvement allowing an enhanced search among the whole population by increasing the size of the initial population.

Accident and emergency rates should be made somewhat relative to prevailing traffic conditions, for example more accidents will occur on congested streets, or streets pushed close to their maximum velocity, that is, there should be a cost function associated with allowing streets to build up congestion, or to permit too great an upper speed-limit. There should be an element of independent accident and emergency propensity per traffic period given that accidents and emergencies occur that are unrelated to traffic volumes.

[2] http://www.dsg.cs.tcd.ie/transport
[3] http://www.trafficlab.utah.edu/

In order to moderate a larger city and identify different areas or similar areas, different path categories could be included. In addition, by modelling each pathway as individual entities, the local clustering of pathways may better represent the dynamic and volatile requirements of urban traffic.

Further to the concept of nodes, it may be desirable to investigate cells in particular cells of varying area/population. This may be based on the concept of activation thresholds in ant/termite colony, e.g. removing workers will demonstrate worker behaviour in soldiers, albeit after a greater exposure to stimulus.

In the current version of the system re-routing does not occur (cars are slowing down, or blocked but do not change route). In the next version of the system, we would want to add a querying capability to the network, i.e. a vehicle can request alternative routes from nodes when progress is stifled.

References

1. London travel report 2005. Technical report, Transport for London, 2005.
2. H. Ceylan and M. G. H. Bell. Traffic signal timing optimisation based on genetic algorithm approach, including driver's routing. *Transportation Research Part B*, 38:329–342, 2004.
3. Y. Ando et al. Pheromone Model: Application to Traffic Congestion Prediction. In *Engineering Self-Organising Systems*, volume 3910 of *LNAI*, pages 182–196. Springer-Verlag, 2005.
4. R. Hoar, J. Penner, and C. Jacob. Evolutionary Swarm Traffic: If Ant Roads Had Traffic Lights. In *Congress on Evolutionary Computation (CEC'02)*, pages 1910–1915. IEEE, 2002.
5. S. Oussedik, D. Delahaye, and M. Schoenauer. Dynamic Air Traffic Planning by Genetic Algorithms. In *Congress on Evolutionary Computation*, pages 1110–1116. IEEE Press, 1999.