Proceedings of the 2007 IEEE
Intelligent Transportation Systems Conference
Seattle, WA, USA, Sept. 30 - Oct. 3, 2007

Decentralised Car Traffic Control using Message Propagation and Re-routing

Martin Kelly, 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 decentralised car traffic control simulation based on re-routing of vehicles and local propagation of messages among traffic nodes (roads intersections and traffic lights). A genetic algorithm is used to identify parameters' values governing the simulations. This paper reports on the one hand on convergence results of the genetic algorithm, and on the other hand on preliminary comparison results of the fittest solutions.**

I. Introduction

Road congestion is a well known problem which affects both rural and urban areas. Different solutions are investigated from ways of increasing capacity of the roads (creating more lanes) to complete traffic management systems combining real-time acquisition of traffic decisions, design of decisions, and propagation of information to drivers. Usually, road congestion is monitored locally through cameras or sensors, traffic flow information is then sent to control centres, where decisions regarding traffic control are taken and the corresponding solutions are then sent back to the vehicles on the road (e.g. through traffic signals, alerting police, etc.). Even though control centres form an indisputable part of traffic management, the density of cars and the complexity of the whole system calls for complementary self-organising or decentralised solutions taking place locally at the roads level.

We report here on an on-going work aiming at deriving different decentralised models and simulations car traffic control based on message propagation among neighbouring road elements such as roads intersection, traffic lights, or cars. The main goal is to maximise traffic throughput and minimise travel time of vehicles. Simulations are controlled by diverse parameters whose values are separately established by the help of a genetic algorithm. In a previous paper we reported on a first model based essentially on local propagation of speed-up and slow-down messages among road intersection [5]. A second model has been defined where messages occur among traffic light controllers and are requests for green or red lights. In addition to message propagation, this second model incorporates as well the notion of re-routing of vehicles when the waiting time is too long. This model and first results related to the convergence of the genetic algorithm up to 158 generations have been reported in [8]. In this paper, on the one hand we report on new convergence results of the genetic algorithm up to 312 generations, yielding to the determination of two local optimum (fittest solutions), as well as on preliminary evaluation of performances of the fittest solutions. On the other hand, we also discuss issues related to self-organisation and control of such systems in an actual environment.

The specificity of our different models lies both in the decentralised approach for traffic control, and in the large number of cars used to simulate traffic (over 15'000). This paper particularly focuses on high a decentralised model for high congestion rates (over 50%).

Section II describes our model, while Section III describes the genetic algorithm used for optimising the model's parameters. Section IV presents results related to the convergence of the genetic algorithm towards the fittest solution, and preliminary performance evaluation of the fittest solution when compared with random cases or put under different conditions (no re-routing, and increased traffic throughput). Section V mentions some related works, while Section VI discusses the feasibility of decentralised approaches in practice.

II. Control Model and Simulation

A. City and Model elements

The city is modelled as a square grid of 20 nodes by 20, representing a 2km*2km city space. The distance between nodes is set at 100 metres. Nodes are road intersections and each node maintains a traffic light controller. A lane is a portion of road between two nodes and has a direction (8 lanes are connected to each node). Even though there is currently no mapping to an actual city, such a grid can be adapted to a large variety of non-square grid cases, by just varying the length of the lanes, and by making some of them one way only.

A whole simulation comprises 15'200 vehicles travelling

permanently in the city streets (corresponding to 50% traffic congestion), and 1'520 different interconnected lanes.

Each simulation starts with 10 vehicles distributed at random across each lane. Each vehicle then randomly targets a destination on the opposite side of the city and generates its optimum route. This provides some routes which are more congested than others from the beginning. The actual speed through the city has a maximum of 30 miles per hour (50km/h). Vehicles try to move at this speed unless there is congestion. The simulation stops after 1'800 seconds of virtual time, i.e. one half-hour of activity.

B. Control and Message propagation

Traffic light controllers can communicate with each other and ask for green or red lights forward or downward lanes. Nodes cycle the traffic light (pass a token) between their inbound lanes, in order to grant them access to all outbound lanes. Cars do not communicate with each other, but have the possibility to re-evaluate their routes when traffic is congested.

1) Parameters

Table 1 summarises the parameters used to control message propagations and re-routing in a decentralised way. Cars, nodes and lanes locally decide (on the basis of these parameters) when to re-route, grant access to roads, or request green lights.

	1	2	3	4	5	6	7
EvaluationTrigger	3%	6%	9%	12%	15%	18%	21%
ResponseThreshold	20%	30%	40%	50%	60%	70%	80%
RequestThreshold	20%	30%	40%	50%	60%	70%	80%
RequestLimit	2	4	6	8	10	12	14
Phase	0m	5m	10m	15m	20m	25m	30m

Table 1: Parameter's Values

EvaluationTrigger: tendency for vehicles to re-evaluate a route which is taking longer than initially planned to resolve. The parameter value represents the % of chances of re-evaluating a journey which is perceived as running late. EvaluationTrigger is between 3% and 21% (3% increment).

ResponseThreshold: propensity of nodes to respond positively to change the traffic signals. The parameter value represents a propensity between 20% and 80% (10% increment) that the target node will observe a response issued from a lane.

RequestThreshold: propensity of lanes to raise a request to change traffic signals. It is a probability curve, with higher congestion leading to greater likelihood of request transmission. Values are the same as the ResponseThreshold parameter.

RequestLimit: sets the number of requests that a node may ignore before a change becomes mandatory. It is an absolute value ranging from 2 to 14 (increment of 2) inclusive for the maximum number of cycles during which a lane's message may be ignored.

Phase: influences the likelihood of a lane to cycle the token when no cars are within range of the intersection. The Phase parameter determines the minimum distance from the intersection the foremost vehicle must be before the lane will ask to cycle the token. The Phase parameter is an absolute value and represents a distance between 0 and 30 metres (increment of 5). If the next vehicle is beyond this distance from the target node (intersection) the lane will raise a request to change the signal.

2) Message Propagation

Messages propagate from lanes to forward and backward nodes. Tokens are employed by nodes to permit an inbound lane access to its outbound lanes. The token (i.e. the green light) is cycled through all inbound lanes one at a time. Lanes employ the use of messages in the simulation in an attempt to influence the node's control of the token. There are three types of message: 1. *forward* messages to *cycle the token* onwards (from green to red or from red to green); 2. *backward* messages to *cycle to token* onwards; and 3. *ignore* messages. Messages are raised in the five following situations:

A lane sends messages *forward* when it detects that vehicles are stationary waiting for the green light, asking the forward node to *cycle its token onwards*, i.e. to rotate the green light towards the lane in order to allow the cars to leave the lane.

A lane may also send messages *backward* when it detects that it is congested, asking the backward node to *cycle its token onwards*, i.e. to rotate the green light towards another lane in order to stop the flow of cars coming in.

When no vehicle are within range of the intersection, the lane may send a *forward* message to the forward node asking it to *cycle the token*, i.e. to change from green light to red because no vehicle is ready to cross the intersection, or to ask for the red light in anticipation of the arrival of the car.

When the vehicle at the head of a lane has a green light but is blocked because the lanes in front are congested, the lane sends a *forward* message to the forward node asking it to *cycle the token*, so that the first car will be prevented to block the intersection.

Finally, a lane may also over-write a previous message at any time according to the latest prevailing conditions: *ignore* messages. This includes cancellation messages to null out any previous change requests.

C. Routes determination

Each of the 15'200 vehicles evaluates its entire route prior to commencing its journey using a variation of the A*-pathfinding algorithm [1].

Whenever the EvaluationTrigger threshold is reached, the journey is likely to be re-evaluated from its current point to the route's endpoint.

Given the current position of the car, the A*-pathfinding algorithm needs to find a new route from that position to the destination point. The best route is the one that minimises the cost of going from the current point to the destination. According to the A*-pathfinding algorithm a route is scored according to the equation F = G + H, where G is the cost to go from the current position of the car to a certain node along the considered route, and H is the *estimated* cost to go from that node to the destination (See [8] for more details on this algorithm). Whenever a car decides to re-evaluate its route (based on the EvaluationTrigger parameter), the city infrastructure calculates the current best route based on local information provided by the different nodes. It then passes this information to the car.

III. Genetic algorithm and Fitness Function

15'200 vehicles permanently populate the streets and constantly undertake journeys during the evaluation period (vehicles start over a new journey when they have completed their current journey). Fitter sets of genes values will permit more journeys to be undertaken during the 1800 seconds of simulation time. Therefore the solution with the most journeys completed is the fittest.

A. Fitness Function

The fitness function is defined as follows:

$$\text{Fitness Function} = \text{journeys}*100 + \text{avgspeed}*10 .$$

The value "journeys" represents the number of completed journeys. The value "avgspeed" represents the average of the speed of all the incomplete journeys. To differentiate between competing solutions (with the same number of completed journey), we also look at the set of current but incomplete journeys.

B. Genetic Algorithm

We evaluate an initial set of 49 individuals. Once that set is evaluated we select candidate genes for reproduction, applying crossover and mutation to yield two child genes. These genes are then run through the simulation. All genomes tested must be unique, i.e. we will not evaluate the same gene sequence more than once. If genes prove fitter than the worst case currently maintained, it is added to the set of fittest candidates and the lowest performer is dropped from the gene-pool. Our tests are concerned with two priorities: 1. to identify the fittest candidate from the simulations undertaken, and 2. to demonstrate convergence towards fitter solutions across the entire gene-pool.

GA Genes. The five genes are EvaluationTrigger, RequestThreshold, ResponseThreshold, RequestLimit, and Phase. They have seven possible values, yielding a search space of 16'807 individual solutions.

GA Initialisation. The GA starts with an initial set of 49 candidates, chosen with middling gene values, at random between 3 and 5 inclusive (see Table 1).

Crossover and Mutation. During reproduction we select two candidate genomes from the genepool using roulette-wheel selection weighted by relative fitness. A random point from the second to fourth gene is selected and two sub-strands extracted from each parent creating two individual siblings. We then subject the siblings to potential mutation. Mutation is set at 10% initially. When an attempt to add an individual fails, we increase the mutation rate by one to help to promote wider selection. After each successful addition to the genepool, mutation is then reset to 10%.

IV. Experiments and Results

As said previously, we reported in [8] results up to 158 generations (for a total of 3315 simulations) obtained by running the genetic algorithm described above. This allowed us to find a first local optimum at generation 133. We continued further with the genetic algorithm, and we extended our results up to 312 generations. We performed in total 8094 simulations. A second local optimum, better than the first one according to the fitness function, has been discovered at generation 271. The results presented here show first how the genetic algorithm converged towards the fittest solution, and how the two local optimums found during the two experiments behave under different conditions. In the rest of this paper we will refer to these local optimum as the 1st and 2nd optimum respectively.

1) Global Result

Table 2 shows at a global level how the two optimum compare with respect to the worst simulation encountered while running the genetic algorithm. We can see that there is about a 13% respectively 16% variation between worst and best case (1st and 2nd optimum) in terms of journeys made.

	Fitness	Journeys
Minimum	2'454'721	24'546
1st Optimum	2'774'920	27'748
2nd Optimum	2'847'821	28'477
Average	2'642'442	26'423
Absolute Difference (1st optimum)	320'199	3'202
Percentage Difference	13.04%	13.04%
Absolute Difference (2nd optimum)	393'100	3'931
Percentage Difference (2nd optimum)	16.01%	16.01%

Table 2: Global Results

2) Number of individuals per generation

Significantly at higher generations, many more simulations had to be undertaken at increasing rates of mutation in order to find candidates fit enough to be added to the genepool.

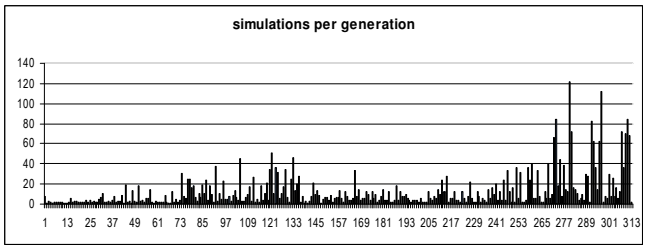

Figure 1: Number of simulations per generation

Figure 1 shows the number of new individuals created at each generation. We can see that for the generations 120-130, shortly before finding the first local optimum at generation 133, a high number of simulations have been performed. Our interpretation is that simulations are stuck in local minima around generation 120-130.

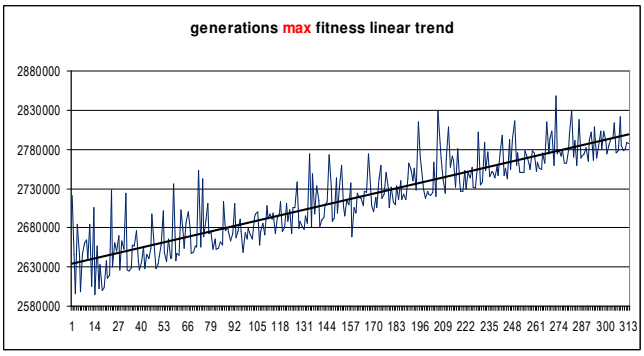

Figure 2: Max fitness function value per generation

Once the local is found, at generation 133, the number of simulations then returns to low numbers until the 2nd optimum is found at generation 271. The number of simulations per generation then increases drastically demonstrating the difficulty of the algorithm to find a better optimum. The number of simulations per generation is in direct proportions to the mutation rate. The mutation rate increases regularly until a solution is found, this allows the system to go out of the local optimum area.

Figure 2 shows two different aspects: first the genetic algorithm converges linearly towards the fittest solutions: the maximum fitness function found at each generation displays a linear trend progression. Second, by comparing Figure 1 and 2, we can see the effect that mutation has on the simulations: each peak in Figure 2 corresponds to a local maximum in Figure 1.

3) Convergence of genetic algorithm

Figure 3 shows the progression of the average fitness function values at each generation (over the different individuals created at each generation). This figure shows that the average fitness value remains rather stable in the first part of the experiment (generations 0-158), i.e. around the 1st optimum we found; it then increases when the 2nd optimum is found (generations 158-312) and remains stable again. The stability effect is due to the fact that there are large differences between minimum and maximum fitness values inside a given generation.

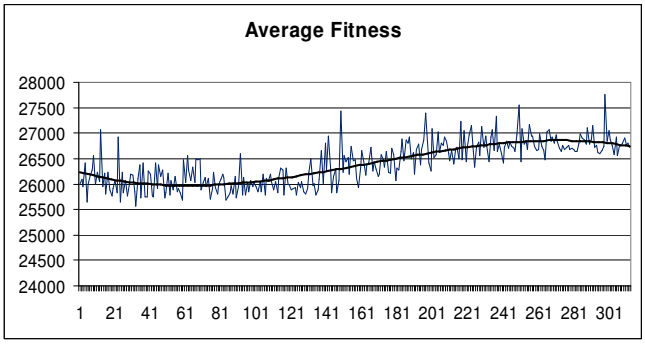

Figure 3: Average fitness function value per generation

4) Genes and Corresponding Parameters values

Table 3 shows the actual values for the parameters defined in Section II.B.

Fitness

Genes	EvalTrigger	ReqThr	RespThr	ReqLimit	Phase	Fitness
Gen 0	15 %	60 %	70 %	10	15m	2721821
Gen 77	18 %	60 %	70%	10	15m	2672821
Gen 156	3 %	50%	30%	10	15m	2736720
Gen 234	3 %	50%	70%	14	15m	2776921
Gen 312	3 %	60%)	50%	12	25m	2668921

Overall

Genes	EvalTrigger	ReqThr	RespThr	ReqLimit	Phase	Fitness
Min (143)	9 %	70%	70%	10	30m	2454721
1st Opt (133)	3 %	70%	70%	10	25m	2774920
2nd Opt (271)	3 %	50%	50%	6	20m	2847821

Table 3: Parameters' values

The first set of rows shows the gene values at different points of our experiments: generation 0 (random genes), generation 77 (middle of first experiments); generation 157 at the end of the first experiment during which the 1st optimum has been found; generation 234 (middle of second series of experiments); and generation 312 (end of genetic algorithm). The second set of rows shows the exact genes values for the least fit individual (found at generation 143); 1st optimum at generation 133, and 2nd optimum at generation 271. Generally a lower tendency to re-evaluate journeys tends to improve fitness (low values for the EvalTrigger gene). In all cases, the Fitness column represents the maximum fitness value obtained for the corresponding generation.

5) Performances of fittest solution

Random solutions. We compared the two optimums against randomly generated routes and randomly generated solutions provided by Generation 0 (see Table 4). Both solutions are significantly better than the average random case (first row

of Table 3). Compared with the maximum random case in Generation 0, the 1st optimum is only slightly better, while the 2nd optimum shows a significant improvement. This is in line with Figure 3 above showing that in average the progression towards fittest solutions occurs by steps and for a given step (0-157) or (157-312) the average values are rather stable and similar.

	Journeys
Gen 0 (avg)	25'987
Gen 0 (max)	27'217
1st Optimum	27'748
2nd Optimum	28'477

Table 4: Comparison with random solutions

Message Propagation only. We slightly modified the simulation program in order to disable completely the re-routing of vehicles (A*-pathfinding) and we ran the simulations for the two optimum we found. We kept the message propagation only. The simulations were governed by the parameters of Table 3 (except for the EvalTrigger, which was disabled).

	Journeys (with re-eval)	Journeys (without re-eval)
1st Optimum	27'748	24'684
2nd Optimum	28'477	25'272

Table 5: No re-routing

As Table 5 shows, the total absence of re-routing causes the 1st optimum to behave as badly as the worst case found (see Table 2, number of journeys for the Minimum case). The 2nd optimum behaves slightly better, but still is much worse than its original version. The EvalTrigger parameter of both 1st and 2nd optimum is 0 (3%), which means that there the propensity of re-evaluation the cars re-routing is very low in fittest solutions (Table 3). As a conclusion of this experiment, we can say that a low re-evaluation rate is very beneficial to traffic control (as fittest solutions show), while no re-routing at all is detrimental.

Higher congestion rates. In a second experiment, we increased the congestion rate from 50% to 75%. The two optimums have been executed with 19'000 vehicles roaming the city (instead of the original 15'200). Table 6 shows that in both cases, the result is similar to the average number of journeys throughout all the simulations (see row Average in Table 2). The explanation for this is that the genes have been optimised for 50% saturation and not 75%.

	Journeys (50% saturation)	Journeys (75% saturation)
1st Optimum	27'748	26'791
2nd Optimum	28'477	26'727

Table 6: 75% Saturation

V. Related Works

Swarm-based traffic control usually employs ant metaphor for inducing a decentralised traffic control. We can cite [2] 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 pheromone later retrieved by other cars provides an indication about traffic congestion and thus serves for short-time traffic predictions. The control model we propose provides both re-routing of vehicles and traffic light cycling. Communication is used only for traffic light cycling. Similarly [4] use 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. This work shares similarities to our model, it couples re-routing and traffic light timing, but uses the ant metaphor as the basic communication mechanism. An ant-based system for decentralised re-routing is provided by [7]. It follows the AntNet algorithm. Artificial ants roam the network of streets and update routing tables at each node (road intersection) which serve for guiding cars. Data provided by cars themselves is also used to enrich the routing.

Decentralised solutions, without message propagation, can be worth mentioning as well. De Oliveira et al. [3] use a reinforcement learning algorithm for updating the parameters of traffic light controllers at run-time in non-stationary environments, specifically studying individual drivers' behaviours. This work focuses on individual traffic light controllers and how they can best learn and adapt themselves to on-going traffic condition. The model we propose does not include the possibility to update or change parameters at run-time - parameters are fixed for a whole simulation run - but considers cooperation among traffic light controllers (through exchange of messages). Rochner et al. [6] use a specific three-layer architecture, where the first layer acts as a "reflex" layer and sits at the level of the traffic light controllers: fixed durations, or variable phases based on traffic detectors information. The second layer is based on monitoring, experience and learning and acts on the parameters of the first layer: identified traffic situations are mapped to parameters of the first layer. The third layer is based on some planning concerns, and uses an internal simulation to help take decisions for unknown situations, by optimising the values of the parameters of the lower layers. The third layer works "off-line" contrarily to the first layer which is for decisions that have to be taken on the fly as congestion arises. This work focuses on individual traffic light controllers and provides a long-term run-time learning

capability.

VI. Discussion

Self-organisation and impact on global traffic. This paper focuses specifically on high congestion rates. It must be noted that the journeys start at one end of the city and finish at the other end. As a result, the centre of the city is particularly congested. Both message propagations and re-routing occur on a *local* basis, lanes locally decide to trigger messages asking for traffic light cycling, nodes locally decide whether or not grant access to such requests, and cars locally decide on whether or not take a new route. Message propagation serves to maximise throughput, by facilitating *cooperation* at junctions and preventing one route from dominating the signal. The messages propagated from lanes to forward and backward nodes help to ameliorate local conditions. Despite the heavily congested nature of the city during rush-hour we facilitate local variations in congestion through this mechanism. Route re-evaluation, occurring when the EvaluationTrigger is set, serves to better *distribute traffic* throughout the city by taking advantage of less-congested laneways. Messages have also an indirect impact on routing and re-evaluations. As traffic builds up, lanes tend to restrict access to themselves, and request access to others.

Feasibility of proposed model. Routing techniques and on-board devices guiding drivers exist already. As we have seen in Section IV, the re-evaluation rate plays a significant role in the performance of the whole system. Therefore, such routing devices should take into account current road activity and be aware of the re-evaluation rate (e.g. it could be obtained when the car enters the city zone controlled by the corresponding parameters). Message propagations among traffic lights controllers would necessitate improved traffic lights controller bound with sensors measuring traffic activity and able to wirelessly communicate with their neighbours (other distant traffic light controllers).

The genes (or parameters) chosen in this model are *propensities*; they are not hard threshold; they indicate a tendency to re-route or to send messages when a certain threshold is reached. Therefore, they do not tend to integrate sudden changes in the system, but smooth adaptations to the current conditions.

Trustworthiness and control. The major drawback or weakness of decentralised/self-organising solutions lies in the current lack of possibilities (in general) to prove or ensure that self-organising or purely decentralised algorithms are actually reliable. Simulations are at the moment the only verifications tool at our disposal; however more formal tools are necessary before decentralised solutions may be accepted and deployed at a large scale in the public. The second major weakness of decentralised solutions is related to top-down control that any traffic management centre would like to impose at some point. Different reasons support the need for high-level control of decentralised system: priority events, resetting the system, etc. When actually implementing and deploying decentralised solutions, hooks for propagation high-level control decisions down to the local nodes should be provided.

VII. Conclusion

This paper proposes a decentralised car traffic control combining both re-routing of vehicles and exchange of messages between lanes and traffic light controllers (asking for traffic light cycling). Focus is on highly congested city centres (over 50% of congestion). Parameters of the system are set up off-line by the use of a genetic algorithm. Results show that a small propensity to re-route vehicles is highly beneficial, i.e. deciding to re-route in 3% of the cases only is already sufficient, while no re-routing at all is detrimental.

Future work will first consist in extending the current exchange of messages by introducing the notion of "cells", where adjacent nodes spontaneously form or dismantle cells when they are in similar or different congestion conditions respectively. Nodes belonging to the same cells have then priority when requesting light cycling. Second, as most simulations of car traffic control, we are using a square grid of routes for modelling the city. Once, the simulation proves to be worthy in this "simplistic" case, it will be necessary to translate it into an actual car traffic schema. Finally, the models proposed consider fixed parameter values that allow the system to adapt to changing conditions within a fixed period of time (e.g. from 8am to 10am). In order to enhance the adaptability to unexpected traffic conditions, it is necessary to integrate into the model the possibility to change these parameters on the fly (e.g. combining instance message propagation with reinforcement learning).

References

[1] http://www.policyalmanac.org/games/aStarTutorial.htm.

[2] 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.

[3] D. de Oliveira et al. "Reinforcement Learning-based Control of Traffic Lights in Non-Stationary Environments: A Case Study in a Microscopic Simulator". In *Fourth European Workshop on Multi-Agent Systems (EUMAS'06)*, 2006.

[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] M. Kelly and G. Di Marzo Serugendo. "A Decentralised Car Traffic Control System Simulation Using Local Message Propagation Optimised with a Genetic Algorithm". *In Engineering Self-Organising Systems*, volume 4335 of LNAI, pages 192--210. Springer-Verlag, 2007.

[6] F. Rochner et al. "An Organic Architecture for Traffic Light Controllers". In *Informatik 2006 - Informatik fur Menschen*, volume P-93 of Lecture Notes in Informatics, pages 120--127. Kollen Verlag, 2006.

[7] B. Tatomir, L. Rothkrantz. "Dynamic Traffic Routing Using Ant Based Control". In *IEEE International Conference on Systems, Man and Cybernetics*, pp. 3970-3975, vol. 4, 2004

[8] M. Kelly and G. Di Marzo Serugendo. "Decentralised Car Traffic Control Simulation Using Message Propagation Optimised with a Genetic Algorithm". In *IEEE Congress in Evolutionary Computation (CEC 2007)*, in press, 2007.