Self-organizing Traffic Lights

Carlos Gershenson*
Centrum Leo Apostel, Vrije Universiteit Brussel,
Krijgskundestraat 33 B-1160 Brussels, Belgium

Steering traffic in cities is a very complex task, since improving efficiency involves the coordination of many actors. Traditional approaches attempt to optimize traffic lights for a particular density and configuration of traffic. The disadvantage of this lies in the fact that traffic densities and configurations change constantly. Traffic seems to be an adaptation problem rather than an optimization problem. We propose a simple and feasible alternative, in which traffic lights self-organize to improve traffic flow. We use a multi-agent simulation to study three self-organizing methods, which are able to outperform traditional rigid and adaptive methods. Using simple rules and no direct communication, traffic lights are able to self-organize and adapt to changing traffic conditions, reducing waiting times, number of stopped cars, and increasing average speeds.

1. Introduction

Anyone living in a populated area suffers from traffic congestion. Traffic is time, energy, and patience consuming. This has motivated people to regulate traffic flow in order to reduce the congestion. The idea is simple: if vehicles are allowed to go in any direction, there is a high probability that one will obstruct another. To avoid this, rules have been introduced to mediate [1] between the conflicting vehicles, by restricting or bounding their behavior. People have agreed on which side of the street they will drive (left or right); traffic lanes prevent cars from taking more space than necessary; traffic signals and codes prompt an appropriate behavior; and traffic lights regulate the crossing of intersections.

There is no solution to the traffic congestion problem when the car density saturates the streets, but there are many ways in which the car flow can be constrained in order to improve traffic. Traffic lights are not the only component to take into account, but they are an important factor. We can say that a traffic light system will be more efficient if, for a given car density, it increases the average speeds of vehicles. This is reflected in less time that cars will wait behind red lights.

For decades, people have been using mathematical and computational methods that find appropriate periods and phases (i.e., cycles) of

*Electronic mail address: cgershen@vub.ac.be; http://homepages.vub.ac.be/~cgershen.

Complex Systems, **16** (2005) 29–53;

https://doi.org/10.25088/ComplexSystems.16.1.29

traffic lights, so that the variables considered will be optimized. This is good because certain synchronization is better than having no correlation of phases. However, many methods applied today do not consider the current state of the traffic. If cars are too slow for the expected average speed, this might result in the loss of the phases dictated by the traffic lights. If they go too fast, they will have to wait until the green light phase reaches every intersection. The optimizing methods are blind to "abnormal" situations, such as many vehicles arriving or leaving a certain place at the same time, such as a stadium, financial district, or university. In most cases, traffic agents need to override the traffic lights and personally regulate the traffic. Nevertheless, traffic modeling has improved greatly our understanding of this complex phenomenon, especially during the past decade [2–7], suggesting different improvements to the traffic infrastructure.

We believe that traffic light control is not so much an optimization problem, but rather an adaptation problem, since traffic flows and densities change constantly. Optimization gives the best possible solution for a given configuration. But since the configuration is changing constantly in real traffic, it seems that we would do better with an adaptive mechanism than with a mechanism that is optimal sometimes, and creates havoc other times. Indeed, modern "intelligent" advanced traffic management systems (ATMS) use learning methods to adapt phases of traffic lights, normally using a central computer [8, 9].[1] Another reason for preferring an adaptive method is that optimization can be computationally expensive. Trying to find all possible optimal solutions of a city is not feasible, since the configuration space is too huge, uncertain, and changes constantly.

In this paper, we present three simple traffic-responsive methods for traffic light control that are adaptive by self-organization, and compare them with two fixed-cycle nonadaptive methods and another traffic-responsive method. We use multi-agent computer simulations to do this. In the next section, we make a brief and practical introduction to the concept of self-organization. Then we present the simulation and the control methods compared. We show our first results in section 5. We present improvements to our simulation to make it more realistic in section 6. The results of further experiments are shown in section 7. We discuss the results and implications in section 8 and conclude in section 9.

[1] A drawback of ATMS is their high cost and complexity that requires maintenance by specialists. There is yet no standard, and usually companies are hired to develop particular solutions for different cities.

2. Self-organization

The term *self-organization* has been used in different areas with different meanings, such as cybernetics [10, 11], thermodynamics [12], mathematics [13], computing [14], information theory [15], synergetics [16], and others (for a general overview, see [17]). However, the use of the term is subtle, since any dynamical system can be said to be self-organizing or not, depending partly on the observer [11, 18].

Without entering into a philosophical debate on the theoretical aspects of self-organization, a practical definition will suffice for our present work. For us, a system described as self-organizing is one in which elements interact in order to achieve a global function or behavior. This function or behavior is not imposed by a single or few elements, nor determined hierarchically. It is achieved dynamically as the elements interact with one another. These interactions produce feedbacks that regulate the system.

Many distributed adaptive traffic light systems can be considered as self-organizing [19, 20]. Nevertheless, the methods presented in this paper distinguish themselves because there is no communication between traffic lights, only local rules (an analysis of their indirect interactions is given in section 8). Still, they are able to achieve global coordination of traffic.

We believe that this approach is useful for systems such as traffic lights, since the "solution" of the problem is not known beforehand, but strived for dynamically by the elements of the system. In this way, systems can adapt quickly to unforeseen changes as elements interact locally. It should be noted that self-organizing approaches are being used in other areas of traffic control [21].

The present work is very abstract. The models presented were not developed to be directly applied on real scenarios (more realistic simulations and pilot studies would be required), but to explore and understand principles of self-organization in traffic light control. The next section describes the simulation where we test various models

3. The simulation

Several traffic simulations use cellular automata to model traffic effectively [20, 22–24], since they are computationally cheap. However, the increase of computing power in the past few years has allowed the development of multi-agent simulations to create more realistic traffic simulations [25–28].

We developed a simulation in NetLogo [29], a multi-agent modeling environment, and we extended the "Gridlock" model [30] which is included in the NetLogo distribution. The Gridlock model consists of an abstract traffic grid with intersections between cyclic (toroidal)

https://doi.org/10.25088/ComplexSystems.16.1.29

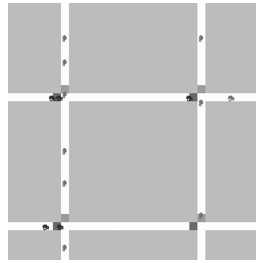

Figure 1. Screenshot of part of traffic grid. Green lights southbound, red light eastbound.

single-lane arteries of two types: vertical or horizontal. Cars are situated initially randomly through the grid with a zero velocity. In the first series of experiments, similar to the scenario of [31], cars only flow in a straight line, either eastbound or southbound. Each crossroad has traffic lights that allow traffic flow in only one of the arteries which intersect it with a green light. Yellow or red lights stop the traffic. The light sequence for a given artery is green-yellow-red-green. Cars try to go at a maximum speed of 1 "patch" per time step, but stop immediately when a red or yellow light is in front of them (one patch) and decrease their speed if there are cars ahead with a slower speed (changing immediately to the speed of the car ahead, and then decelerating). Time is discrete, but not space. Cars use an acceleration of 0.099 to speed up or slow down. A patch is a square of the environment the size of a car. A screenshot of the environment can be seen in Figure 1. The reader is invited to test the simulation (source code included), with the aid of a Java-enabled Internet browser, at [32].

The user can change different parameters, such as the number of arteries or number of cars. Different statistics are shown: the number of stopped cars, the average speed of cars, and the average waiting times of cars.

4. The control methods

4.1 Marching control

This is a very simple method. All traffic lights "march in step:" all green lights are either southbound or eastbound, synchronized in time. Intersections have a phase φ_i, which counts time steps. φ_i is reset to zero when the phase reaches a period value p. When $\varphi_i == 0$, red lights turn green, and yellow lights turn red. Green lights turn yellow one time step earlier, that is, when $\varphi == p - 1$. A full cycle of an intersection consists of $2p$ time steps. "Marching" intersections are such that $\varphi_i == \varphi_j, \forall i, j$.

4.2 Optim control

This method is implemented trying to set phases φ_i of traffic lights in such a way that, as soon as a red light turns green, a car stopped by this would find the following traffic lights green. In other words, we obtain a fixed solution so that green waves flow to the southeast. This green wave method is very popular for avenues that have preference over crossing streets.

The simulation environment has a "radius" of r square patches, so that these can be identified with coordinates (x_i, y_i), $x_i, y_i \in [-r, r]$. Therefore, each artery consists of $2r+1$ patches. In order to synchronize all the intersections, red lights should turn green and yellow lights should turn red when

$$\varphi_i == \text{round}\left(\frac{2r + x_i - y_i}{4}\right) \tag{1}$$

and green lights should turn to yellow the previous time step. The period should be $p = r + 3$. The three is added as an extra margin for the reaction and acceleration times of cars (found to be best for low densities, by trial and error, for the parameters used in the experiments discussed in section 5).

A disadvantage of the optim control is that the average speed decreases as the traffic density increases, so cars cannot keep up the speed of the green waves. A different solution could be obtained, for lower average speeds, but then the green waves would be too slow for low traffic densities.[2]

These two first methods are nonadaptive, in the sense that their behavior is dictated beforehand, and they do not consider the actual state of the traffic.

4.3 Sotl-request control

All three self-organizing control methods use a similar principle: traffic lights keep a counter κ_i which is set to zero when the light turns red and then incremented at each time step by the number of cars approaching only the red light (i.e., the next one a car will reach) independently of the status or speed of the cars (i.e., moving or stopped). When κ_i reaches a threshold θ, the green light at the same intersection turns yellow, and the following time step it turns red with $\kappa_i = 0$, while the red light which counted turns green. In this way, if there are more cars approaching or waiting behind a red light, this will turn into green faster than if there are only a few cars. This simple mechanism achieves self-organization in the following way: if there is one or a few cars, they will

[2]Some real traffic light systems have different "optimal" solutions (i.e., different p and φ_i values) for different times of the day [8].

https://doi.org/10.25088/ComplexSystems.16.1.29

be stopped for more time behind red lights. This gives time for other cars to join them. As more cars join the group, they will wait less time behind red lights. With a sufficient number of cars, the red lights will turn green even before they reach the intersection, generating "green corridors." Having "platoons" or "convoys" of cars moving together improves traffic flow, compared to a homogeneous distribution of cars, since there are large empty areas between platoons, which can be used by crossing platoons with less interference.

The sotl-request method has no phase or internal clock. Traffic lights change only when the given conditions are met. If there are no cars approaching a red light, the complementary one can stay green. However, depending on the value of θ, high traffic densities can trigger the lights to switch too fast, obstructing traffic flow.

4.4 Sotl-phase control

The sotl-phase method differs from sotl-request adding the following constraint: A traffic light will not be changed if the number of time steps is less than a minimum phase, that is, $\varphi_i < \varphi_{\min}$ (φ_i is the number of time steps since the light turned green). Once $\varphi_i \geq \varphi_{\min}$, the lights will change when $\kappa_i \geq \theta$. This prevents the fast switching of lights.[3]

4.5 Sotl-platoon control

The sotl-platoon method adds two further restrictions to sotl-phase to regulate the size of platoons. Before changing a red light to green, it checks if a platoon is not crossing through, in order not to break it. More precisely, a red light is not changed to green if on the crossing street there is at least one car approaching within ω patches from the intersection. This keeps crossing platoons together. For high densities, this restriction alone would cause havoc, since large platoons would block the traffic flow of intersecting streets. To avoid this, we introduce a second restriction. Restriction one is not taken into account if there are more than μ cars approaching the intersection. With sotl-platoon, long platoons can be broken, and the first restriction only comes into play if a platoon will soon be through an intersection.

We say that these three adaptive methods are self-organizing because the global performance is given by the local rules followed by each traffic light: they are unaware of the state of other intersections and still manage to achieve global coordination.

The sotl methods use a similar idea to the one used by Porche and Lafortune in [34, and references within], but with a much simpler implementation. There is no costly prediction of arrivals at intersections,

[3] A similar method has been used successfully in the United Kingdom for some time, but for isolated intersections [33].

and no need to establish communication between traffic lights to achieve coordination. They do not have fixed cycles.

4.6 Cut-off control

We wanted to compare our self-organizing methods with a traditional traffic-responsive method, that has proven to be better than static methods at single intersections [35]. The idea of the cut-off method is simple: a traffic light will remain green until a queue of stopped waiting cars reaches a length of λ cars. At this moment, the green light turns yellow, and at the next time step, red, while the opposing light turns green. The advantage of this method is that it can adapt to changing traffic demands. The disadvantage is that cars need to stop at an intersection before it changes.

Recall that sotl methods keep a count of approaching cars, independently of their speed. Therefore, cars do not need to stop in order to change a traffic light.

4.7 No-corr control

To have an idea of the benefit of the different control methods, we also compared them with a noncorrelated scheme no-corr: each traffic light is assigned a phase φ_i at random, and its value does not change during a simulation run. They all have the same period p. Thus, there is no correlation between different intersections.

5. First results

We performed simulations in order to obtain average statistics on the performance of the different control methods. These were namely speed,[4] percentage of stopped cars, and waiting time. The results shown in Figures 2 and 3 were obtained from runs of 10,000 time steps with random initial conditions in a grid of 10×10 arteries of $r = 80$ (therefore 3120 available patches), with $p = 83$, $\theta = 41$, $\varphi_{\min} = 20$, $\omega = 4$, $\mu = 3$, and $\lambda = 3$. These parameters were found to be the best for each method by a trial-and-error exploration of the parameter space. For each method we made one run varying the number of cars from 20 to 2000, in steps of 20 (101 runs in total), with the same parameters. The results were extracted from NetLogo using a small Java program.[5]

We can see that the marching method is not very efficient for low traffic densities, that is, when there are roughly less than three cars between

[4]The cruise speed is 1 patch/time step, that is, the speed at which cars go without obstructions.

[5]The precise figures were taken from the total averages for each variable (black lines in the simulation plots).

https://doi.org/10.25088/ComplexSystems.16.1.29

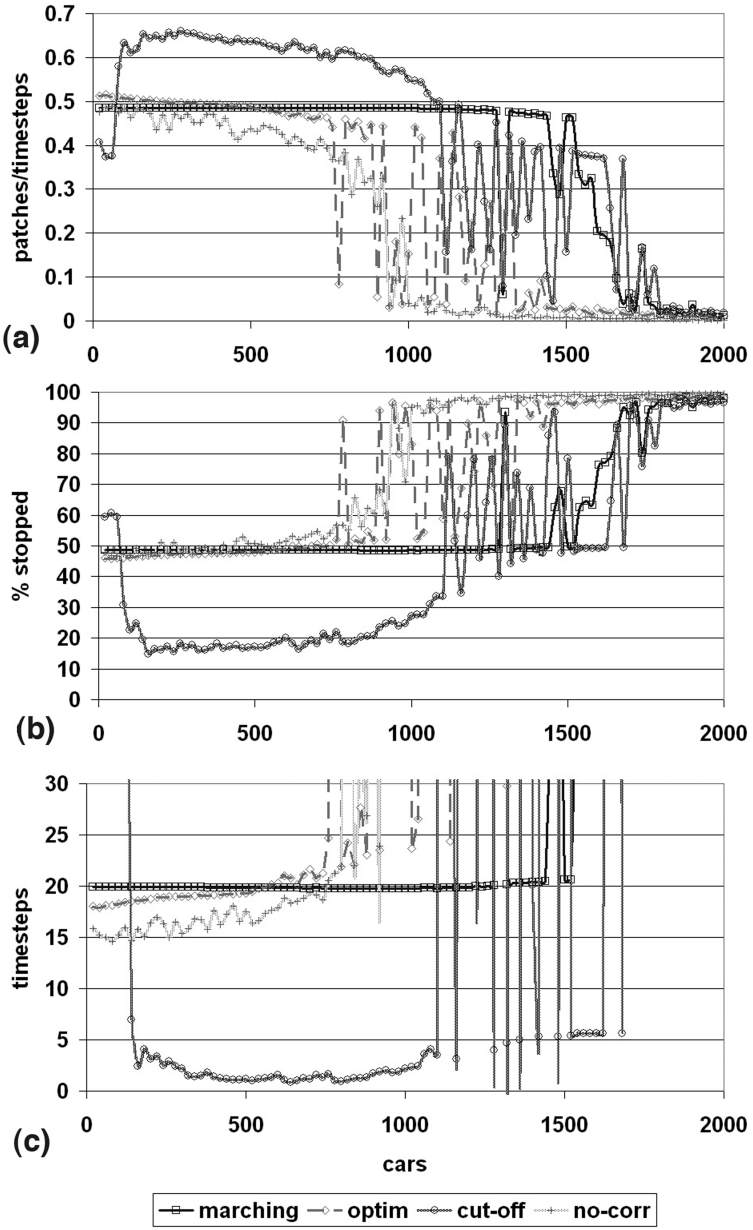

Figure 2. Results for standard methods. (a) Average speeds of cars. (b) Percentage of stopped cars. (c) Average waiting times. Very high waiting times (out of graph) indicate deadlocks.

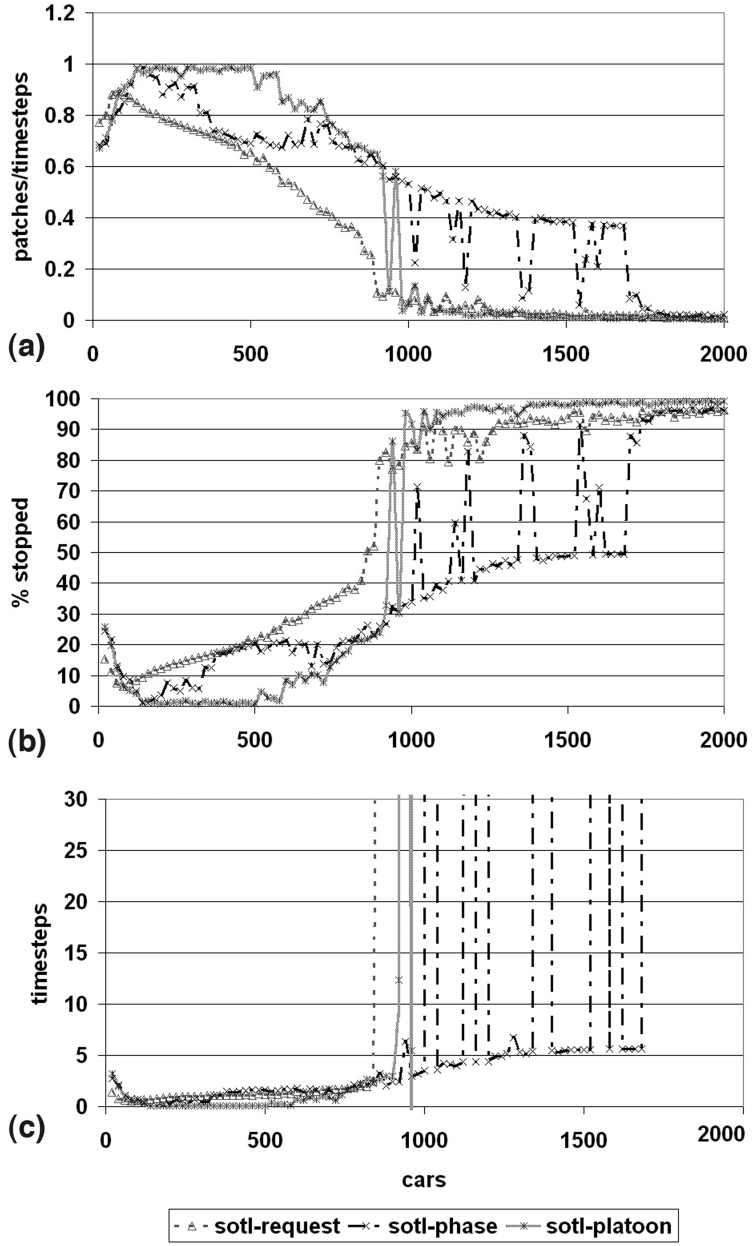

Figure 3. Results for self-organizing methods. (a) Average speeds of cars. (b) Percentage of stopped cars. (c) Average waiting times. Very high waiting times (out of graph) indicate deadlocks.

https://doi.org/10.25088/ComplexSystems.16.1.29

intersections. Since half of the arteries (all eastbound or all southbound) have red lights, this causes almost half of the cars to be stopped at any time, reducing the average speed of cars. On the other hand, its performance degrades slowly as the traffic densities reach certain levels, and performs the best for very high densities, that is, when more than eight cars are encountered between intersections, and traffic jam formation is probable. This is because it keeps a strict division of space occupied by cars, and interferences are less probable. Still, when there are too many cars, "deadlocks" are formed, that is, all cars are stopped.

For low densities, the optim method performs acceptably. However, for high densities cars can enter a deadlock much faster than with other methods. This is because cars waiting behind other cars at red lights do not reach green waves, reducing their speed and the speed of the cars which go behind them. Also, even when there will be some cars that do not stop, flowing through green waves, there will be an equivalent number of cars waiting to enter a green wave, losing the time gained by cars in green waves. Therefore, the performance cannot be much better than marching.

Sotl-request gives the best performance for low traffic densities because platoons can quickly change red lights into green, in most cases before actually reaching the intersections. Since the traffic density is low, this does not obstruct many cars approaching the intersection in the corresponding artery. However, for high densities this method is extremely inefficient, since there is a constant switching of lights due to the fact that θ is reached very fast. This reduces the speed of cars, since they stop on yellow lights, but also breaks platoons, so that the few cars that pass will have a higher probability of waiting more time at the next intersection.

Sotl-phase does not perform as good as sotl-request for low densities because in many cases cars need to wait behind red lights as κ_i reaches $\varphi_{\min}$, with no cars coming in the corresponding artery. The performance of sotl methods could be improved for low densities by reducing θ, since small platoons might need to wait too long at red lights. As the traffic density reaches a medium scale, platoons effectively exploit their size to accelerate their intersection crossing. With the considered parameters, in the region around 160 cars, and again at around 320, sotl-phase can achieve full synchronization in space, in the sense that no platoon has to stop, so all cars can go at a maximum speed. (In the graphs, the average speed reflects the average time it takes to achieve full synchronization, i.e., closer to one is faster.) This is not a realistic situation, because synchronization is achieved due to the toroidal topology of the simulation environment. Still, it is interesting to understand the process by which the full synchrony is reached. Platoons are formed, as described in the previous section, of observed sizes $3 \leq \text{cars} \leq 15$. One or two platoons flow per street. Remember that platoons can change red lights

to green before they reach an intersection, if $\kappa_i \geq \varphi_{\min}$. If a platoon moving in an artery is obstructed, this will be because still $\kappa_i < \varphi_{\min}$, and because a platoon is crossing, or crossed the intersection recently in the complementary artery. The waiting of the platoon will change its phase compared to other flowing platoons. However, if no platoon crossed recently, a platoon will keep its phase relative to other platoons. This induces platoons not to interfere with each other, until all of them go at maximum speed. We can see that this condition is robust by resetting the traffic light periods and κ_i. Each reset can be seen in the spikes of the graphs shown in Figure 4. Nevertheless, the precise time at which full synchronization is reached can vary. For some initial conditions, full synchronization is not achieved, but it is approached nevertheless.

The phenomenon of full synchronization shows us how self-organizing traffic lights form platoons, which in turn modulate traffic lights. This feedback is such that it maximizes average speeds and minimizes waiting times and stopped cars in a robust way. The self-organizing traffic lights are efficient without knowing beforehand the locations or densities of cars.

When there is a very high traffic density, optim and sotl-request reach deadlocks frequently, where all traffic is stopped. Sotl-phase behaves similar to marching, since traffic lights change as soon as $\kappa_i \geq \varphi_{\min}$, because in most cases $\kappa_i \geq \theta$ by then. This also reduces the sizes of platoons, which if very long can generate deadlocks. However, when the traffic density is too high, deadlocks will be inevitable, though marching generates less deadlocks than sotl-phase. This is because with the marching method whole arteries are either stopped or advancing. This reduces the probability of having a green light where cars cannot cross (e.g., due to a red light ahead, and a line of cars waiting to cross it), which would block the crossing artery at the next phase.[6]

Sotl-platoon manages to keep platoons together, achieving full synchronization commonly for a wide density range, more effectively than sotl-phase. This is because the restrictions of this method prevent platoons from leaving a few cars behind, with a small time cost for waiting vehicles. Still, this cost is much lower than breaking a platoon and waiting for separated vehicles to join back again. A platoon is divided only if $\mu = 3$, and a platoon of size three will manage to switch traffic lights without stopping for the simulation parameters used. However, for high traffic densities platoons aggregate too much, making traffic jams more probable. The sotl-platoon method fails when a platoon waiting to cross a street is long enough to reach the previous intersection, but not long enough to cut its tail. This will prevent waiting cars from

[6]Deadlocks could be avoided by restricting all cars to cross intersections only if there is at least one free space after it. However, it is unrealistic to expect human drivers to behave in this way.

https://doi.org/10.25088/ComplexSystems.16.1.29

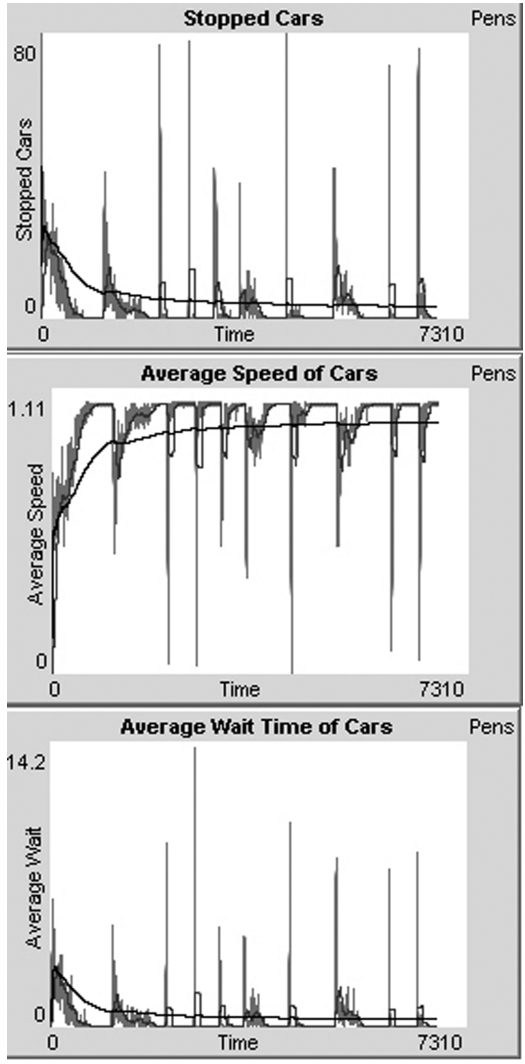

Figure 4. Resets of traffic lights as sotl-phase achieves full synchronization (80 cars in a 5×5 grid with $r = 40$).

advancing, until more cars join the long platoon. This failure could probably be avoided by introducing further restrictions in the method, but here we would like to study only very simple methods.

The platoon size in sotl strategies depends on the tolerance θ and the distance between crossings, since longer distances give more time for κ_i to reach θ. An alternative would be to count cars at a specified distance, independently of the distance between crossings, so that the

method could also be useful when traffic lights are very close together, or far away. This should also be considered in a nonhomogeneous grid.

Cut-off performs better than the static methods, as it responds to the current traffic state (except for very low densities, when cars in streets may never reach the cut-off length λ). However, it is not as efficient as sotl methods, since cars need to stop before being able to switch a red light to green. Still, for high densities its performance is comparable to that of sotl-phase, performing better than the other two sotl methods.

With no-corr, we can observe that all the methods have an improvement over random phase assignation. Nevertheless, the difference between no-corr and static methods is less than the one between static and adaptive methods. This suggests that, for low traffic densities, adaptation is more important than "blind" correlation. For high traffic densities, the opposite seems to be the case. Still, adaptive methods have correlation built in.

We performed tests with "faulty," that is, noncorrelated intersections. All methods are robust to failure of synchronization of individual traffic lights, and the global performance degrades gracefully as more traffic lights become faulty.

6. Improvements to the simulation

In order to ensure that the encouraging results of the sotl methods presented in section 5 were not an artifact of the simplicity of the simulation, we made some improvements to make it more realistic. It was good to have a simple environment at first, to understand better the basic principles of the control methods. However, once this was achieved, more complexity was introduced in the simulation to test the performance of the methods more thoroughly. We developed thus a scenario similar to the one of [20].

We introduce traffic flow in four directions, alternating streets. That is, arteries still consist of one lane, but the directions alternate: southbound-northbound in vertical roads, and eastbound-westbound in horizontal roads. Also, we introduce the possibility of having more cars flowing in particular directions. This allows us to simulate peak hour traffic, regulating the percentages of cars that will flow in vertical roads, eastbound, or southbound roads.[7]

The most unrealistic feature of the first simulation was the torus so we introduce an option to switch it off. Cars that exit the simulation are removed from it. For creating new cars, gates are chosen randomly, with a probability proportional to the parameters that represent car percentages at vertical, eastbound, and southbound roads. Then, at

[7] %horizontal = 100 – %vertical; %westbound = 100 – %eastbound; %northbound = 100 – %southbound.

https://doi.org/10.25088/ComplexSystems.16.1.29

chosen gates, a car will be created with a probability

$$P_{\text{newc}} = 1 - \frac{c}{c_{\max}} \tag{2}$$

where c is the current number of cars, and $c_{\max}$ is the maximum number of cars. Note that without a torus, traffic jams are less probable, since new cars cannot be fed into the system until there is space. Therefore, the actual number of cars will be less than $c_{\max}$.

We also add a probability of turning at an intersection P_{turn}. Therefore, when a car reaches an intersection, it will have a probability P_{turn} of reducing its speed and turning in the direction of the crossing street. This can cause cars to leave platoons, which were more stable in the first series of experiments.

7. Second results

We performed similar sets of experiments as those presented earlier. We did runs of 10,000 time steps with random initial conditions in a grid of 10×10 arteries of $r = 80$, with $p = 83$, $\theta = 41$, $\varphi_{\min} = 20$, $\omega = 4$, $\mu = 3$, and $\lambda = 3$. The percentage of cars in horizontal streets was the same as in vertical, but of those, 60% in vertical roads were southbound (40% northbound) and 75% in horizontal streets were eastbound (25% westbound). We used $P_{\text{turn}} = 0.1$. Since each street crosses 10 other streets, on average each car should turn more than once. Results of single runs, increasing the number of initial cars ($c_{\max}$ in equation (2)) from 20 to 2000 in steps of 20, can be appreciated in Figures 5 and 6. We should note that the average number of cars is reduced as the initial density increases, since cars cannot enter the simulation until there is space for them. This reduces considerably the probability of deadlocks. We can see a plot comparing the initial and average number of cars for the simulations in Figure 7.

In general terms, the improvements of the simulation did not alter the first results by much. Marching and optim are poor for low traffic densities, but the performance degrades smoothly as the density increases. There are almost no deadlocks because with high densities inserted in the simulation more cars exit than enter. If this was a real city, there would be queues waiting to enter the city, which the statistics of our simulations do not consider.

Sotl-request performs the best for low traffic densities, but worst for high densities, even worse than no-corr. This is because, as in the first results, dense platoons force the traffic lights into a constant switching, which reduces the performance.

The method sotl-phase avoids this problem with the restriction set by $\varphi_{\min}$. It still performs very good for low densities, and the average speed degrades slowly to a comparable performance with the nonadaptive

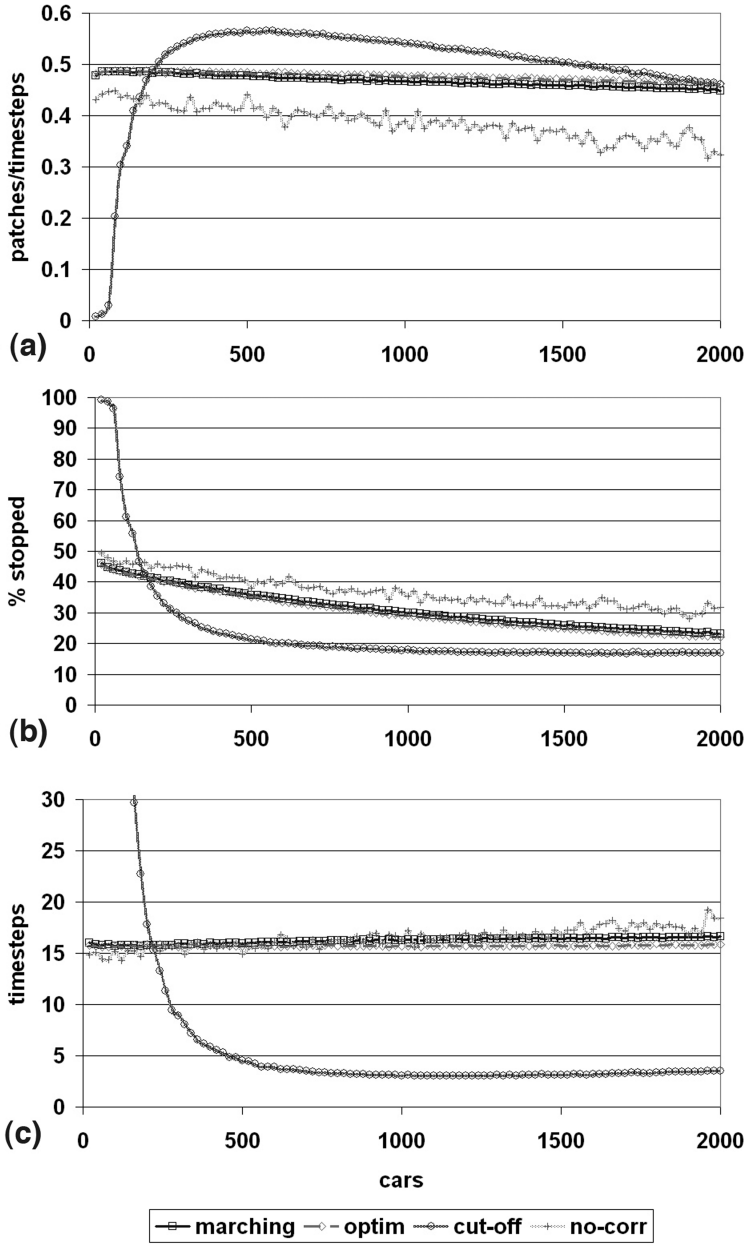

Figure 5. Results in four directions, turning, and without torus, for standard methods. (a) Average speeds of cars. (b) Percentage of stopped cars. (c) Average waiting times.

https://doi.org/10.25088/ComplexSystems.16.1.29

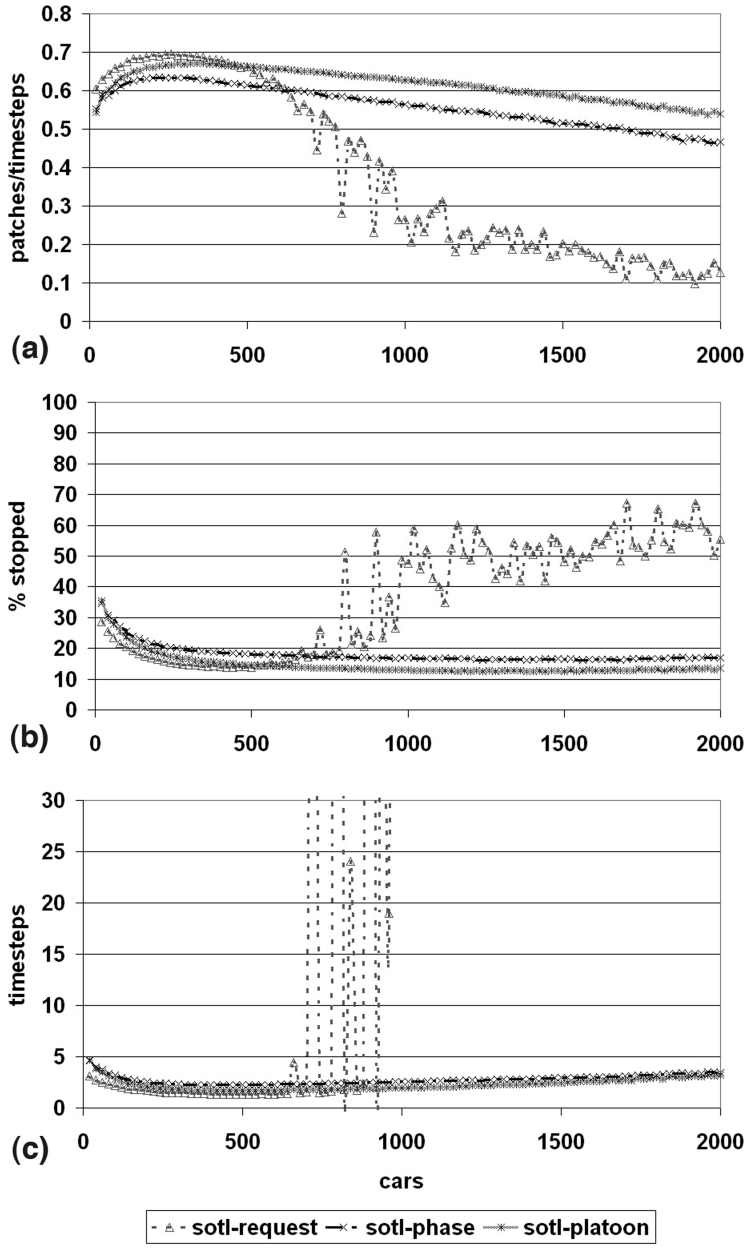

Figure 6. Results in four directions, turning, and without torus, for self-organizing methods. (a) Average speeds of cars. (b) Percentage of stopped cars. (c) Average waiting times.

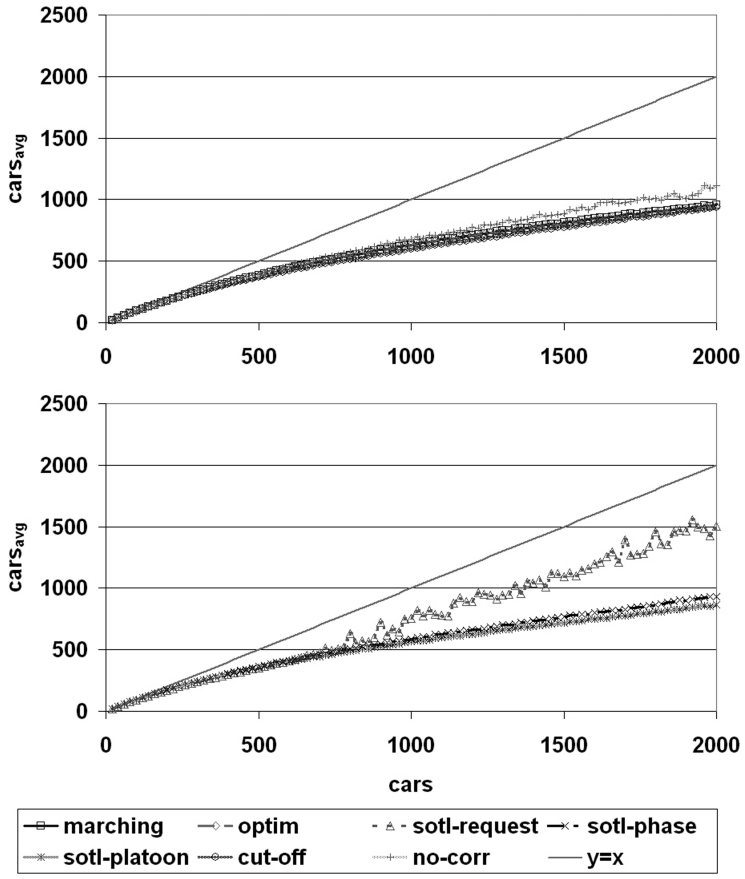

Figure 7. Comparison of initial and average number of cars for different methods without torus.

methods. However, the percentage of stopped cars and the waiting times are much lower than the nonadaptive methods.

Sotl-platoon manages to keep platoons together, which enables them to leave the simulation faster. It gives on average 30% (up to 40%) more average speed, half the stopped cars, and seven times less average waiting times than nonresponsive methods. Therefore, this method performs the best overall. It can adapt to different traffic densities, minimizing the conflicts between cars. It is not possible to achieve almost perfect performance, as it did for medium densities with a torus, since cars enter the simulation randomly. Still, this method is the one that manages to adapt as quickly as possible to the incoming traffic, effectively organizing vehicles into platoons that quickly leave the simulation, even when single vehicles might break apart from them (due to $P_{\text{turn}} > 0$).

https://doi.org/10.25088/ComplexSystems.16.1.29

The cut-off method again performs badly for very low densities. Still, afterwards it performs better than the nonadaptive methods, but not as good as sotl-phase or sotl-platoon.

Again, no-corr shows that all methods give an improvement over random phase assignment, except for sotl-request at high densities, where the method clearly breaks down.

The average number of cars, shown in Figure 7, can be taken as an indirect measure of the methods' performance: the faster the cars are able to leave the simulation, there will be less cars in it, thus more efficient traffic flow. We can observe an inverse correlation between the average number of cars and the average speeds. If the traffic lights can "get rid" of the incoming traffic as quickly as possible, it means that they are successfully mediating the conflicts between vehicles.

The phenomenon of full synchronization is destroyed if there is no torus, or if $P_{\text{turn}} > 0$. However, it is still achieved when the cars flow in four directions, or when the number of horizontal arteries is different from the number of vertical arteries. It is easier to reach if there are less arteries in the simulation. Also, if the length of horizontal and vertical arteries differs, that is, $r_x \neq r_y$, full synchronization is more difficult to obtain, since the periods of the platoons passing the same traffic light depend on the length of the arteries. If these are proportional, for example, $r_x = 2r_y$, full synchronization can be achieved. Nevertheless, the sotl-phase and sotl-platoon methods achieve very good performance under any of these conditions.

8. Discussion

The series of experiments performed show that sotl strategies are more efficient than traditional control methods. This is mainly because they are "sensitive" and adaptive to the changes in traffic. Therefore, they can cope better with variable traffic densities, noise, and unpredicted situations. Based on our results, we can say the following.

- The formation of platoons can be seen as a reduction of variety [36, Ch. 11]. It is much easier to regulate 10 groups of 10 cars than 100 cars independently.[8] Platoons make the traffic problem simpler. Oscillations in traffic will be reduced if cars interact as groups. We can also see this as a reduction of entropy: if cars are homogeneously spread on the street grid, at a particular moment there is the same probability of finding a car on a particular block. This is a state of maximum entropy. However, if there are platoons, there will be many blocks without any car, and a few with several. This allows a more efficient distribution of resources,

[8]This could be seen as "functional" modularity [37, pp. 188–195].

namely free space at intersections.[9] It is interesting to note that the sotl methods do not force vehicles into platoons, but induces them. This gives the system flexibility to adapt.

- We can say that the sotl methods try to "get rid" of cars as fast and just as possible. This is because they give more importance to cars waiting for more time compared to recent arrivals, and also to larger groups of cars. This successfully minimizes the number of cars waiting at a red light and the time they will wait. The result is an increase in the average speeds. Also, the prompt "dissipation" of cars from intersections will prevent the formation of long queues, which can lead to traffic jams.

- Since cars share a common resource—space—they are in competition for that resource. Self-organizing traffic lights are synergetic [16], trying to mediate conflicts between cars. The formation of platoons minimizes friction between cars because they leave free space around them. If cars are distributed in a homogeneous way in a city, the probability of conflict is increased.

- There is no direct communication among the self-organizing traffic lights. However, they "exploit" cars to stigmergically transmit information,[10] in a way similar to social insects exploiting their environment to coordinate. For traffic lights, car densities form their environment. Traffic lights respond to those densities. But cars also respond to the traffic light states. We could say that traffic lights and cars "co-control" each other, since cars switch traffic lights to green, and red traffic lights stop the cars.

8.1 Adaptation or optimization?

Optimization methods are very effective for problems where the domain is fairly static. This enables the possibility of searching in a defined space. But in problems where the domain changes constantly, such as traffic, an adaptive method should be used, to cope with these changes and constantly approach solutions in an active way.

The problem of traffic lights is such that cars and traffic lights face different situations constantly, since they affect each other in their dynamics (i.e., traffic lights affect cars, cars affect cars). With sotl methods, cars affect traffic lights and traffic lights affect other traffic lights stigmergically via the cars. If the situation is unknown or unpredictable, it is better to use an adaptive, self-organizing strategy for traffic lights, since it is not computationally feasible to predict the system behavior.[11]

[9] The formation of platoons has already been proposed for freeways, with good results [38].

[10] For an introduction to stigmergy, see [39].

[11] This is because there is a high sensitivity to initial conditions in traffic, that is, chaos: if a car does not behave as expected by a nonadaptive control system, this can lead the state of the traffic far from the trajectory expected by the system.

https://doi.org/10.25088/ComplexSystems.16.1.29

We can see an analogy with teaching: a teacher can tell exactly a student what to do (as an optimizer can tell a traffic light what to do). But this limits the student to the knowledge of the teacher. The teacher should allow space for innovation if some creativity is to be expected. In the same way, a designer can allow traffic lights to decide for themselves what to do in their current context. Stretching the metaphor, we could say that the self-organizing traffic lights are "gifted with creativity," in the sense that they find solutions to the traffic problem by themselves, without the need of the designer even understanding the solution. On the other hand, nonadaptive methods are "blind" to the changes in their environment, which can lead to a failure of their rigid solution.

We can deduce that methods which are based on phase cycles, and even adaptive cyclic systems [9, 19] (i.e., systems that try to coordinate phases with fixed durations) will not be able to adapt as responsively as methods that are adaptive and noncyclic, since they are not bounded by fixed durations of green lights [34]. Therefore, it seems that optimizing phases of traffic lights is not the best option, due to the unpredictable nature of traffic.

All traffic lights can be seen as mediators [1] among cars. However, static methods do not take into account the current state of vehicles. They are more "autocratic." On the other hand, adaptive methods are regulated by the traffic flow itself. Traffic controls itself, mediated by "democratic" adaptive traffic lights.

8.2 Practicalities

There are many parallel approaches trying to improve traffic. We do not doubt that there are many interesting proposals that could improve traffic, for example, to calculate real-time trajectories of all cars in a city depending on their destination via GPS. However, there are the feasibility and economic aspects to take into account. Two positive points in favor of the self-organizing methods is that it would be very easy and cheap to implement them. There are already sensors on the market which could be deployed to regulate traffic lights in a way similar to sotl-phase. Sensors implementing the sotl-platoon method would not be too difficult to deploy. Moreover, sotl methods could be introduced gradually in a city, adapting to the existing network. The system does not need to be implemented completely to start working and giving results. Secondly, there is no need of a central computer, expensive communication systems, or constant management and maintenance. The methods are robust, so they can resist incrementally the failure of intersections.

Self-organizing traffic lights would also improve incoming traffic to traffic light districts, for example, from freeways, since they adapt actively to the changing traffic flows. They can sense when more cars are coming from a certain direction, and regulate the traffic equitably.

Pedestrians could be included in a self-organizing scheme by considering them as cars approaching a red light. For example, a button could be used, as is now common, to inform the intersection, and this would contribute to the count κ_i.

Vehicle priority could also be implemented, by simply including weights w_j associated to vehicles, so that the count κ_i of each intersection would reach the threshold θ counting $w_j c * ts$. However, this would require a more sophisticated sensing mechanism, although available with current technology for priority vehicle detection. Still, this would provide an adaptive solution for vehicle priority, which in some cities (e.g., London) can cause chaos in the rest of the traffic lights network, since lights are kicked off phase.

We should also note that traffic lights are not the best solution for all traffic situations. For example, roundabouts [40] are more effective in low speed, low density neighborhoods.

8.3 Unattended issues

The only way of being sure that a self-organizing traffic light system would improve traffic is to implement it and find out. Still, the present results are encouraging to test our methods in more realistic situations.

A future direction worth pursuing would be a systematic exploration of the parameters θ, p, and $\varphi_{\min}$ values for different densities, as well as the exploration of different environmental parameters. A meta-adaptive method for regulating these parameters depending on the traffic densities would be desirable, but preliminary results have been discouraging. In real situations this could be easier, because the efficiency of different values can be tested experimentally for specified traffic densities. Therefore, if a certain density is detected, proper parameter values could be used. More realistic situations should also be added to our simulations, such as multiple-street intersections, multiple-lane streets, lane changing, different driving behaviors, and nonhomogeneous streets. It would also be interesting to compare our methods with others, for example, [9, 19], but many of these are not public, or very complicated to implement in a reasonable amount of time. Reinforcement learning methods [26] will adapt to a particular flow density. However, in real traffic densities change constantly and unevenly. We should compare the speed of adaptation of these methods with the proposed self-organizing ones, but intuition tells us that learning methods will be effective only for a particular fixed traffic density. We would also like to compare our methods with other distributed adaptive cyclic methods, for example, [20, 41] (sotl and cut-off are noncyclic), to test if indeed phase cycles reduce the adaptability of traffic lights.

Another direction worth exploring would be to devise methods similar to the ones presented that promote "optimal" sizes of platoons for

https://doi.org/10.25088/ComplexSystems.16.1.29

different situations. We would need to explore as well which platoon sizes yield less interference for different scenarios.

9. Conclusions

We have presented three self-organizing methods for traffic light control which outperform traditional methods due to the fact that they are "aware" of changes in their environment, and therefore are able to adapt to new situations. The methods are very simple: they give preference to cars that have been waiting longer, and to larger groups of cars. Still, they achieve self-organization by the probabilistic formation of car platoons. In turn, platoons affect the behavior of traffic lights, prompting them to turn green even before they have reached an intersection. Traffic lights coordinate stigmergically via platoons, and they minimize waiting times and maximize average speeds of cars. Under simplified circumstances, two methods can achieve robust full synchronization, in which cars do not stop at all.

From the presented results and the ones available in the literature [34], we can see that the future lies in schemes that are distributed, noncyclic, and self-organizing. In the far future, when autonomous driving becomes a reality, new methods could even make traffic lights obsolete [42, 43], but for the time being, there is much to explore in traffic light research.

There are several directions in which our models could be improved, which at the present stage might be oversimplifying. However, the current results are very promising and encourage us to test self-organizing methods in real traffic environments.

Acknowledgments

Ricardo Barbosa, Vasileios Basios, Pamela Crenshaw, Kurt Dresner, Peter Furth, Francis Heylighen, Bernardo Huberman, Stuart Kauffman, Tom Lenaerts, Mike McGurrin, Kai Nagel, Marko Rodriguez, Andreas Schadschneider, Seth Tisue, Bart de Vylder, and one anonymous referee provided useful comments and assistance in the development of this manuscript. This research was partially supported by the Consejo Nacional de Ciencia y Teconolgía (CONACyT) of Mexico.

References

[1] Francis Heylighen, "Mediator Evolution: A General Scenario for the Origin of Dynamical Hierarchies," submitted, 2004.

[2] I. Prigogine and R. Herman, *Kinetic Theory of Vehicular Traffic* (Elsevier, New York, 1971).

[3] D. E. Wolf, M. Schreckenberg, and A. Bachem, editors, *Traffic and Granular Flow '95* (World Scientific, Singapore, 1996).

[4] M. Schreckenberg and D. E. Wolf, editors, *Traffic and Granular Flow '97* (Springer, Singapore, 1998).

[5] D. Helbing, H. J. Herrmann, M. Schreckenberg, and D. E. Wolf, editors, *Traffic and Granular Flow '99: Social, Traffic, and Granular Dynamics* (Springer, Berlin, 2000).

[6] Dirk Helbing, *Verkehrsdynamik* (Springer, Berlin, 1997).

[7] Dirk Helbing and Bernardo A. Huberman. "Coherent Moving States in Highway Traffic," *Nature*, **396** (1998) 738–740.

[8] Federal Highway Administration, *Traffic Control Systems Handbook* (U.S. Department of Transportation, February 1998).

[9] P. B. Hunt, D. I. Robertson, R. D. Bretherton, and R. I. Winton, "SCOOT: A Traffic Responsive Method of Coordinating Signals," Technical Report, TRRL, 1981.

[10] Heinz von Foerster. "On Self-organizing Systems and Their Environments," in *Self-Organizing Systems*, edited by M. C. Yovitts and S. Cameron (Pergamon, 1960).

[11] W. Ross Ashby, "Principles of the Self-organizing System," in *Principles of Self-Organization*, edited by H. Von Foerster and G. W. Zopf, Jr. (Pergamon, 1962).

[12] G. Nicolis and I. Prigogine, *Self-Organization in Non-Equilibrium Systems: From Dissipative Structures to Order Through Fluctuations* (Wiley, 1977).

[13] G. G. Lendaris, "On the Definition of Self-organizing Systems," *Proceedings of the IEEE*, March 1964.

[14] Francis Heylighen and Carlos Gershenson, "The Meaning of Self-organization in Computing," *IEEE Intelligent Systems*, July/August 2003, 72–75.

[15] Cosma R. Shalizi, "Causal Architecture, Complexity and Self-Organization in Time Series and Cellular Automata," Ph.D. thesis, University of Wisconsin at Madison, 2001.

[16] H. Haken, Synergetics and the Problem of Self-organization," in *Self-Organizing Systems: An Interdisciplinary Approach*, edited by G. Roth and H. Schwegler (Campus Verlag, 1981).

[17] Francis Heylighen, "The Science of Self-organization and Adaptivity," in *The Encyclopedia of Life Support Systems* (EOLSS Publishers, 2003).

https://doi.org/10.25088/ComplexSystems.16.1.29

[18] Carlos Gershenson and Francis Heylighen, "When Can We Call a System Self-organizing?" in *Advances in Artificial Life, Seventh European Conference, ECAL 2003 LNAI 2801*, edited by W. Banzhaf, T. Christaller, P. Dittrich, J. T. Kim, and J. Ziegler (Springer-Verlag, 2003).

[19] A. G. Sims, "SCATS: The Sydney Co-ordinated Adaptive System," in *Proceedings of the Engineering Foundation Conference on Research Priorities in Computer Control of Urban Traffic Systems*, 1979.

[20] Baldo Faieta and Bernardo A. Huberman, "Firefly: A Synchronization Strategy for Urban Traffic Control," Technical Report SSL-42, Xerox PARC, Palo Alto, 1993.

[21] Lars Wischhof, Andre Ebner, Hermann Rohling, Matthias Lott, and Rüdiger Halfmann, "SOTIS: A Self-organizing Traffic Information System," in *Proceedings of the Fifty-seventh IEEE Vehicular Technology Conference* (Jeju, South Korea, 2003).

[22] Ofer Biham, A. Alan Middleton, and Dov Levine, "Self-organization and a Dynamical Transition in Traffic-flow Models," *Physical Review A*, **46** (1992) R6124–R6127.

[23] K. Nagel and M. Schreckenberg, "A Cellular Automaton Model for Freeway Traffic," *Journal of Physics I France*, **2** (1992) 2221–2229.

[24] Debashish Chowdhury and Andreas Schadschneider, "Self-organization of Traffic Jams in Cities: Effects of Stochastic Dynamics and Signal Periods," *Physical Review E*, **59** (1999) R1311–R1314.

[25] Kai Nagel, *Multi-Agent Transportation Simulation* (book in progress, 2005).

[26] M. Wiering, J. Vreeken, J. Van Veenen, and A. Koopman, "Simulation and Optimization of Traffic in a City," in *IEEE Intelligent Vehicles Symposium, (IV'04)* (IEEE, 2004).

[27] Luis Miramontes Hercog, "Co-evolutionary Agent Self-organization for City Traffic Congestion Modeling," in *Genetic and Evolutionary Computation: GECCO 2004: Genetic and Evolutionary Computation Conference, Seattle, WA, USA, June 26–30, 2004. Proceedings, Part II* (Springer-Verlag, 2004).

[28] Danko A. Roozemond and Jan L. H. Rogier, "Agent Controlled Traffic Lights," in *ESIT 2000: European Symposium on Intelligent Techniques* (September 2000).

[29] Uri Wilensky, NetLogo, 1999; http://ccl.northwestern.edu/netlogo.

[30] U. Wilensky and W. Stroup, "NetLogo HubNet Gridlock Model," 2002; http://ccl.northwestern.edu/netlogo/models/HubNetGridlock.

[31] E. Brockfeld, R. Barlovic, A. Schadschneider, and M. Schreckenberg, "Optimizing Traffic Lights in a Cellular Automaton Model for City Traffic," *Physical Review E*, **64** (2001) 056132.

[32] http://homepages.vub.ac.be/~cgershen/sos/SOTL/SOTL.html.

[33] R. A. Vincent and C. P. Young, "Self Optimising Traffic Signal Control Using Microprocessors: The TRRL MOVA Strategy for Isolated Intersections," *Traffic Engineering and Control*, **27**(7–8) (1986) 385–387.

[34] Isaac Porche and Stéphane Lafortune, "Adaptive Look-ahead Optimization of Traffic Signals," *Intelligent Transportation Systems Journal*, **4**(3) (1998).

[35] M. Ebrahim Fouladvand, Zeinab Sadjadi, and M Reza Shaebani, "Optimized Traffic Flow at a Single Intersection: Traffic Responsive Signalization," *Journal of Physics A: Mathematical and General*, **37** (2004) 561–576.

[36] W. Ross Ashby, *An Introduction to Cybernetics* (Chapman and Hall, London, 1956).

[37] Herbert A. Simon, *The Sciences of the Artificial*, third edition (MIT Press, 1996).

[38] Shahab Sheikholeslam and Charles A. Desoer, "Combined Longitudinal and Lateral Control of a Platoon of Vehicles: A System Level Study," Technical Report 1991-09-01, California PATH, 1991.

[39] Guy Theraulaz and Eric Bonabeau, "A Brief History of Stigmergy," *Artificial Life*, **5**(2) (1999) 97–116.

[40] M. Ebrahim Fouladvand, Zeinab Sadjadi, and M. Reza Shaebani, "Characteristics of Vehicular Traffic Flow at a Roundabout," *Physical Review E*, **70** (2004) 046132.

[41] Toru Ohira, "Autonomous Traffic Signal Control Model with Neural Network Analogy," in *Proceedings of InterSymp '97: Ninth International Conference on Systems Research, Informatics and Cybernetics*, Baden-Baden, Germany, August 1997. SCSL-TR-97-004.

[42] Carlos Gershenson, "Control de tráfico con agentes: CRASH," in *Memorias XI Congreso Nacional ANIEI*, Xalapa, México, 1998.

[43] Kurt Dresner and Peter Stone, "Multiagent Traffic Management: A Reservation-based Intersection Control Mechanism," in *The Third International Joint Conference on Autonomous Agents and Multiagent Systems*, 2004.

https://doi.org/10.25088/ComplexSystems.16.1.29