Urban Traffic Control Based on Learning Agents

Pierre-Luc Grégoire, Charles Desjardins, Julien Laumônier and Brahim Chaib-draa
DAMAS Laboratory, Computer Science and Software Engineering Department, Laval University

Abstract— **The optimization of traffic light control systems is at the heart of work in traffic management. Many of the solutions considered to design efficient traffic signal patterns rely on controllers that use pre-timed stages. Such systems are unable to identify dynamic changes in the local traffic flow and thus cannot adapt to new traffic conditions. An alternative, novel approach proposed by computer scientists in order to design adaptive traffic light controllers relies on the use of intelligents agents. The idea is to let autonomous entities, named agents, learn an optimal behavior by interacting directly in the system. By using machine learning algorithms based on the attribution of rewards according to the results of the actions selected by the agents, we can obtain a control policy that tries to optimize the urban traffic flow. In this paper, we will explain how we designed an intelligent agent that learns a traffic light control policy. We will also compare this policy with results from an optimal pre-timed controller.**

I. Introduction

With the increase of vehicular traffic observed in recent years in urban areas, there has been a significant degradation of the efficiency of the traffic flow. This degradation is particularly important in areas where many roads connect. Intersections play a major role in decreasing the flow as controllers managing their traffic lights must cope with irregular traffic flow and with numerous external events that can possibly decrease the efficiency of the flow. This often renders impossible the use of pre-timed controllers for efficient control of the system.

One of the solutions to this problem would be to design controllers that use adaptive policies. Such adaptive systems could react to current perceptions of traffic conditions and select the best actions in order to optimize the traffic flow at the intersection. Moreover, these adaptive systems could even be equipped with communication networks that could enable adaptive coordination between different intersections in order to improve the traffic flow globally. Such coordination could help minimize the overall delay caused by traffic signals that vehicles get by navigating through the urban road network.

Researchers from different fields of study have focused on finding solutions to optimize the efficiency of the traffic flow. Among them, computer scientists were particularly interested in these solutions based on adaptive policies. To design such systems, they have turned to the abstraction of intelligent software agents since this model naturally adapts itself to traffic control systems. With this abstraction, traffic lights can be viewed as entities that act autonomously in order to reach a certain goal. Moreover, these adaptive systems could display social capabilities as they could be extended to controllers that promote coordination and collaboration between intersections in order to manage traffic signals in the most efficient way possible. This could optimize the flow through the network at a larger scale than only locally, at single intersections.

Since most control tasks are often too complex to be modeled and solved exactly by humans, we are interested in letting agents learn their own adaptive behavior by reinforcement learning. Using this approach, agents receive reward signals in response to taking actions in their environment. Learning algorithms then proceed to associate the current state of the agent to the action it should take to maximize its expected reward.

In this paper, we will focus on a single adaptive agent traffic light controller. We will detail the design of this agent and we will illustrate how we use machine learning algorithms to let it learn an adaptive behavior that selects actions according to current observations of local traffic conditions. Finally, we will compare the learned controller with an optimal pre-timed controller.

II. Related Work

In this section, we will briefly survey some of the work related to traffic signal control and some of the work related to the application of intelligent agents to traffic signal control.

First, single-junction traffic signal controllers fall into two categories: fixed-time controllers and traffic-response controllers. Fixed-time controllers generally use various traffic patterns repeated in cycles every day. For example, in this category, [1] have proposed to optimize a single-junction controller using integer and linear programming in order to minimize delay. As for traffic-response control, [2] have proposed a controller that makes dynamical adjustments of cycle time and stage split.

Contrary to single-junction controllers are the centralized traffic control methods. Among these methods are the TRANSYT, SCATS and SCOOT methods [3]. [4] has studied and evaluated the SCOOT method, which controls and optimizes traffic lights. More precisely, the performance of this method is calculated using the bandwidth, the average queue length and the number of stopped vehicles. Data is collected in real time from sensors and, from the updated traffic model, SCOOT makes some modifications on the

This research is funded by the AUTO21 Network of Centers of Excellence, an automotive research and development program focusing on issues relating to the automobile in the 21st century. AUTO21 is a member of the Networks of Centers of Excellence of Canada program. Web site: www.auto21.ca

previous coordination plans. This method has been shown to reduce delay compared to fixed-time traffic lights in England.

A few researchers in computer science have also studied the use of intelligent agents and learning for traffic light control. For example, our work is quite similar to that of [5]. One difference is that Wiering used model-based reinforcement learning to learn traffic light control strategies. His approach was based on a highly simplified model of traffic flow. The author also experimented with co-learning, where cars would share the same value functions as the traffic lights and try to learn the optimal path to their destination.

[6] has tackled the problem differently and has suggested the use of a reservation system for collision avoidance at an intersection. In this system, vehicles make a request to a central agent, the intersection. If the request is accepted, the vehicle needs to follow the prescribed path, and is guaranteed to be safe while crossing the intersection. Conflicts are solved by the intersection, and vehicles that cannot fulfill a suggested plan must stop at the intersection. The idea is interesting, yet is not quite scalable to the presence of human-driven vehicles. It also lacks some coordination aspects between multiple traffic lights, which is one of the most important problem that plagues the design of such controllers.

Coordination of multiple traffic light agents has also been explored by [7], where evolutionary game theory has been applied to let agents learn how to cooperate. This approach needs the presence of a communication system for agents to share their local actions. With the help of [8] the author also focused on the non-stationary aspects of the problem. In order to detect changes in the traffic flow, the authors designed a context detection algorithm that learns and triggers different control policies according to the traffic flow detected. This technique seems particularly useful in order to deal with the fact that different traffic flows are often observed according to certain periods, for example, rush hours both in the morning and at the end of the day. The algorithm was able to detect those changing phases and adapt to currently observed conditions.

Thorpe, through several reports ([9], [10]) has focused on applying learning on a number of traffic lights placed on a grid, much like the approach we suggested here. Through his work, he has illustrated the different issues that are faced (and that we have observed in our experimentations) in the development of such an adaptive traffic control system based on reinforcement learning and autonomous agents.

Finally, other artificial intelligence techniques, such as dynamic programming, have also been used to optimize the coordination of traffic lights. [11] has shown a reduced delay using this technique, particularly when demand reaches the junction's capacity.

III. Traffic Light Control

The present section will describe the inner workings of traffic light signal control. These definitions were used and implemented in our simulator in order to test our learning agent. The exact implementation of traffic light control signals we have considered for our simulations will be described in section V-A.

The first aspect to consider in traffic light control is the description of the possible actions that vehicles can legally make at an intersection. Traffic light controllers possess a traffic signal plan, which gives the different traffic stages available. A traffic stage consists of the description, for every lane of an intersection, of the possible transitions that can be done by vehicles [3]. Traffic stages also give explicitly how lights are displayed for vehicles on the intersection, indicating if they can turn left and/or right when a light is green or stop when it is red. Obviously, the transitions allowed in each stage must be designed without any conflicts. For large junctions, this design task is normally done by a traffic expert assisted by software.

There are two different ways to control the switching between the stages of a signal plan. Fixed-timed controllers are intuitively simple and work on a 'time-of-the-day' basis. This is the most affordable control technique and is the most logical choice for networks with stable or predictable traffic. The counterpart is that their efficiency can be extremely poor when the traffic flow is not stable. The traffic flow may becomes unstable for many reasons like accidents, weather condition, etc. In these cases, other actuated controllers can make use of buried detectors on waiting lanes of an intersection. These controllers are often necessary in these conditions or when the traffic flow does not have predictable patterns.

Intersections also need to have structures that manage and automatizes the control of the lights. The first generation of control structure uses non-computerized systems that are based on hard-wired logic to control the shifting between signal plans. Second generation structures use a centralized computer to control the set of traffic lights at an intersection. By this mean, a single computer can be used to control many intersections[4]. Finally, the third generation of controllers use distributed computers. In that case, individual intersections provide by their own processing unit [7].

Another aspect of traffic light simulation is the synchronisation in arterials. The goal of coordinated systems is to synchronize the traffic signals along an arterial in order to allow vehicles, traveling at a given constant speed, to cross the arterial without stopping at red lights [4]. This is known as a green wave and is normally used at rush hours. To achieve its goal, the designer sets parameters such as the offset between lights and the desired speed of vehicles. The offset is the time between the beginning of the green stage of consecutive traffic lights. To optimize the green wave, the problem is to find the largest time period during which a vehicle is able to continue without stopping at any intersection if it maintains the wanted fixed speed. This time period is known as the bandwidth.

Finally, many metrics exist in order to evaluate the performance of a traffic light controller. One of the most important metric to optimize, as it impacts directly on drivers is called the delay. The delay is defined by the amount of additional

time a vehicle takes to complete its journey through the network because of traffic lights [3]. Another interesting metric is the throughput which gives the number of vehicles that cross the intersection in a specified amount of time. Clearly, the general optimization goal for traffic network designers is to lower the delay and to increase the throughput of vehicles.

The adaptive controllers we propose are actuated controllers that have a certain perception of their local environment. Through sensors, we suppose they can have a basic perception of the number of vehicles that are present near the intersection. This assumption is critical for our control agents to display reactive behavior. The agent metaphor also implies that we focus on decentralized controllers each having their own computational unit. Finally, the assumptions on the signal plans we use will be described later since the signal plans are closely related to the experiments we make.

IV. Agents and Reinforcement Learning

A. Autonomous Agents

The autonomous agents abstraction is a modern approach in artificial intelligence research that has been developped as an efficient alternative for the resolution of complex problems in many domains of application. By definition, agents are autonomous entities that can act independently in their environment in order to reach a defined goal. Moreover, agents are defined by their ability for social interaction, which enables collaboration and coordination with other agents. Further information about the agent abstraction can be found in [12].

Recently, research in artificial intelligence has focused on using agents for machine learning and thus on formal mathematical aspects of complex decision-making. More precisely, the focus is now on finding optimal policies to solve hard problems where agents might have to act in complex environments that might only be partially observable or even to interact and coordinate with other agents.

The present paper will focus on the use of intelligents agents acting as traffic light controllers. Their task will be to optimize the traffic flow by learning a control policy that chooses when to change signal stages according to their local perception of the environment, which is, in this case, a measure of the quantity of vehicles waiting at the intersection.

B. Reinforcement Learning

To enable deliberation, intelligent agents need a formal framework for decision-making. The past few years have seen the prevalence of the use of the Markov Decision Processes in the field of reinforcement learning. MDPs are efficient at representing sequential decision problems where the goal is to find the best actions to take to maximize utility an agent can receive [13]. Many reinforcement learning algorithms use this framework as a computational model.

Formally, a MDP is defined by:

- A a finite set of actions,
- S a finite set of states,
- $\mathcal{R} : \mathbf{S} \times \mathbf{A} \rightarrow \mathbf{R}$ an immediate reward function,
- $\mathcal{P} : \mathbf{S} \times \mathbf{A} \times \mathbf{S} \rightarrow [0, 1]$ the transition probabilities from a state to another when taking an action.

To find the optimal solution of an MDP, the Markov property must be satisfied. This property indicates that the transition probability of reaching the next state s' only depends on the current state s. With this property, the current state of the system encapsulates all knowledge required to make a decision. To solve an MDP, the value of a state, $V^*(s)$, must be calculated. This value represents the expected reward that the agent can obtain if it executes an optimal policy π^* from state s. To compute the optimal policy, we need to know the values of the states under this optimal policy. This is given by the Bellman optimality equation 1:

$$V^*(s) = \max_a \sum_{s'} \mathcal{P}^a_{ss'} \left[\mathcal{R}^a_s + \gamma V^*(s')\right] \quad (1)$$

This equation can only be used when the dynamics of the system, represented by its transition probabilities $\mathcal{P}^a_{ss'}$, are known. When the agent does not have knowledge of the these probabilities, it must act and observe the outcome of its actions directly in the environment. In that case, the agent has to use state-action values, $Q(s, a)$, that represent the expected utility of taking action a in the state s. Using this definition, it becomes possible to learn the success or failure of actions in a specific state. The Q-Learning algorithm, as seen in the update equation 2, is often used to update the state-action values:

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha \Big[r_{t+1} + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t) \Big] \quad (2)$$

This equation is used by the Q-Learning algorithm [14] and updates the current Q-value using the immediate reward, r_{t+1} and the Q-value of the next state, $Q(s', a)$. The Q-values are guaranteed to converge to the optimal values if this update is done infinitely often for each state action pair. Convergence also depends on α, the learning rate [13].

The downside of this algorithm is that it faces what is called the 'curse of dimensionality'. This curse refers to the fact that the size of the state space (the number of $Q(s, a)$ pairs) can grow exponentially with the number of variables contained in the states and with the number of possible actions. This renders convergence nearly impossible for complex problems. Moreover, the use of a Q-values table means that continuous environments cannot be treated and need to be discretized.

Policy gradient algorithms can address some of these issues. Instead of updating the value function in order to obtain the optimal function, policy gradient algorithms work by keeping an estimated value function. These algorithms use this function estimator to compute a stochastic policy. They modify this estimated function directly in the direction of the gradient of the policy value. The problem becomes

one of estimating the gradient of the policy. The advantages of these learning algorithms is that they can treat continuous states using their estimated value function and that there is no growth of the state space possible since the values of all state-action pairs is given by the function estimator.

We refer the reader to the following papers, [15] and [16], as they describe in further detail this family of learning algorithms.

V. Simulation

We present in this section the simulation environment and the agent learning algorithms that we implement in order to learn a single road junction traffic light signal control policy.

A. Urban Traffic Control Simulator

To design our controllers, we implement in a home-made vehicle simulator a traffic light simulation module. This module features a road network that is made of one long segment crossed by four roads, thus giving a total of four intersections in the system. Each road segment between intersections is 280 meters long. Horizontal roads have two lanes while the vertical road have four lanes.

Vehicles are generated according to a uniform distribution, with a given probability at each step that is fixed for the duration of a simulation. When created, vehicles receive a path to follow. Possible paths are to travel on a road from its beginning to its end. Hence, vehicles on vertical road have to cross four intersections before arriving at their destination while vehicles on horizontal roads have to cross one intersection. On each road, vehicles can go in both direction. Vehicles are created to reach a cruising velocity of $14\ m/s$. Their acceleration is $2\ m/s^2$ which means that they reach their cruising speed in 7 seconds. Their deceleration is $4\ m/s^2$. Each vehicle is driven by a basic controller that keeps a safe distance of two seconds between other vehicles. This controller also reacts automatically to changing traffic lights. For roads where vehicles can choose between two lanes, the controller chooses a lane randomly at the begining of its course and keeps it until the end.

Traffic lights are located at the four road intersections. They each possess a controller that shifts through the different stages of the light in order to direct the traffic flow. The signal plans we use are composed of two stages. As seen in Figure 1, the first signal plan lets vehicles travel in the North/South direction, while the other plan enables traveling in the East/West direction. Each stage is followed by a mandatory 10 timesteps for a yellow light. Timesteps correspond to 0.250 seconds. This light gives enough time for vehicles that are engaged in the intersection to get to the next segment, while letting enough time for incoming vehicles to stop in order to respect the following red light. The only simplification of the system we made was to disallow left or right turns at the intersections.

At each decision step, controllers choose whether they need to shift the light or not for the next stage. The yellow light stage cannot be bypassed, as security would degrade and some collisions could not be avoided. The set of actions

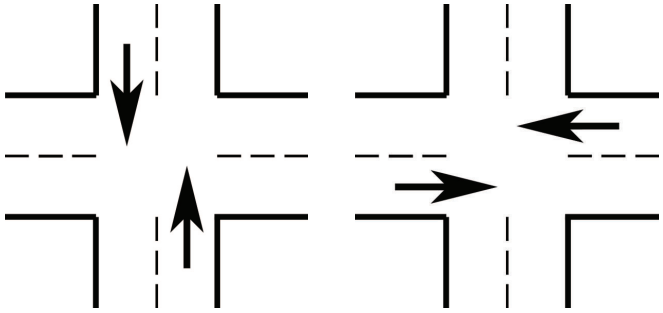

Fig. 1. North/South Stage and East/West Stage

is thus limited to two: a shift action and an action that keeps the current plan. Since decision steps occur at every 50 timesteps, the minimal stage time of a green light is 40 timesteps (if the controller choose the shift action, the first 10 timesteps are for the yellow light). During their travel, vehicles are delayed if they encounter red lights. As soon as a vehicle starts to brake due to a traffic light, it start to compute delay. At each timestep, it updates its delay according its current speed relatively to its desired speed, as seen in equation 3 and 4.

$$D_{t+1}(v) = D_t(v) + d(v) \tag{3}$$

where

$$d(v) = 0.250 * (1 - (S_{c_t}/S_{d_t})); \tag{4}$$

Since our simulation timestep is fixed to 250 milliseconds per timestep, this equation gives the delay in seconds. By weighting the update of each step with the current speed (S_{c_t}) and the desired speed (S_{d_t}), we take into consideration the acceleration and deceleration of the vehicle in its delay.

At each timestep, we can compute the total delay by adding all the vehicles' delays. We get a good performance metric of the overall traffic light network's delay by taking the average delay for all vehicles that are created in a simulation.

As seen in Figure 2, we have used a viewer to directly observe the results of our simulation on the traffic flow.

Fig. 2. Urban Traffic Control Simulator

B. Agent Learning

For our agent to learn a control policy, we first have to define the variables that describe the state of the agent that will be used by the learning algorithm. The variables we consider are the number of horizontal vehicles waiting, the number of vertical vehicles waiting, the current state of the light (if it is green horizontally or vertically) and the number of decision steps since a shift occurred.

The reward function we use gives negative rewards for delayed vehicles. The value of the reward is given by the following function:

$$R(s_t, a_t) \quad = \quad -\sum((d(v)/50)^2) * 1.5 \qquad (5)$$

where $d(v)$ is the delay of vehicle v in seconds. The constant (50) is used to represent an acceptable delay for a vehicle at a junction while the square function and 1.5 constant decreases the reward given when a vehicle waits past the acceptable delay.

In order to let the light learn a behavior that does not shift at each decision step, we added to this reward function a cost for changing the green light's direction. Thus, the agent had to learn to optimize the cost of shifting the light against the cost of vehicle delay.

To learn the policy, our agent uses a policy gradient reinforcement learning algorithm, OLPOMDP, as detailed in section IV-B. In our implementation, we have used a neural network comprised of 4 inputs (corresponding to the 4 state variables), 2 layers of 20 hidden nodes and 2 outputs (corresponding to the possible actions), with inputs being normalized to $[0, 1]$. Since the policy obtained using OLPOMDP is a stochastic policy, the action chosen when executing the policy is the action that has the highest probability.

VI. Results

First, to learn a control policy, we put the traffic light control agent in the simulator, using the reinforcement learning framework described in the previous section. Learning took place in a scenario where there was only one traffic light to optimize. There was a 14 percent probability of generating a vehicle at each step for the North/South direction and a 3 percent probability for the East/West direction. The learning results for this situation are shown in Figure 3. It illustrates the decrease in vehicle delay with the number of episodes of the policy gradient algorithm. These results were obtained from a single run of the algorithm.

Then, we test this learned policy against pre-timed controllers. The first scenario uses a distribution of 14 percent probability of generating a vehicle at each step for the North/South direction and 3 percent probability for the East/West direction. These values are the same as for the learning scenario and give respectively a flow of 1008 and 432 vehicles per hour per lane. Since the capacity of each intersection is of 1400 vehicles per hour per lane, the scenario values are just above the maximum capacity of an intersection.

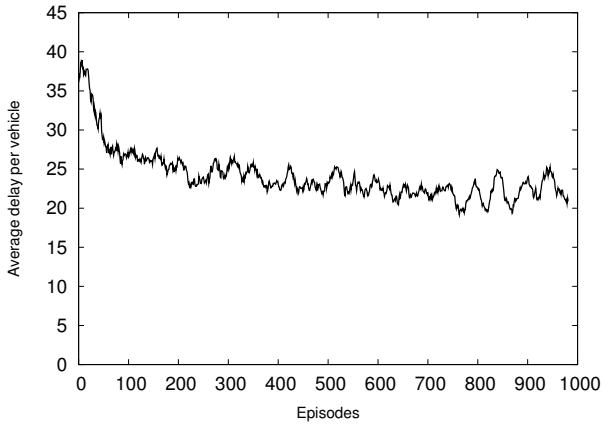

Fig. 3. Policy gradient algorithm learning

Pre-timed controllers are constrained to the same stage time as the learning controller which means that their cycle length must be a multiple of 50 timesteps. The best choice that fits the vehicles generation flow is a cycle length of 150 timesteps with a North-South green stage of 100 timesteps. This give 66% of green time in North/South direction and 33% in East-West direction while the vehicle flow repartition is of 70%/30%.

We first run the simulation on a single road junction signalization, the same that was used to learn the traffic light policy. We compute the average delay per vehicle for the pre-timed controller and the learned policy controller. For each controller, we run 50 episodes to get a representative average result. For comparison, we also run the simulation with a pre-timed sub-optimal controller which allow 50% of green light in both direction.

System	Avg delay	ST DEV
Pre-timed near-optimal	22.10	3.86
Pre-timed sub-optimal	61.59	5.33
Learned	19.96	4.54

TABLE I

Average and standard deviation for vehicle delay for one intersection scenario (Calculated over 50 simulations)

As we can see in Table I, the traffic light agent controller learns a near-optimal policy. The slightly lower performance of the optimal pre-timed controller can be explained by the choice of stage length which in this case is not completely adjusted (but really close) to the generated vehicle flow.

We then run simulations on the four intersections scenario described in section V-A. We use the same flow generation as in the single junction simulation but we generate less vehicles in the South-North direction that in the North-South direction (504 vehicles per hour per lane compare to 1008 vehicles). The pre-timed controllers have been adjusted for a green wave effect, with an offset of 80 timestep. This offset is the required time for a vehicle running at 14 m/s to reach a junction from an other junction (280 meters between each

junction). In Table II, we can see that the controller running the learned policy obtains similar results that the pre-timed synchronized controller.

System	Avg delay	ST DEV
Pre-timed near-optimal	34.86	4.82
Learned	38.45	4.74

TABLE II

AVERAGE AND STANDARD DEVIATION FOR VEHICLE DELAY FOR A FOUR INTERSECTIONS (CALCULATED OVER 50 SIMULATIONS)

The last scenario was to test controllers behavior in presence of changes in traffic flow. We supposed that an external event modified the traffic flow as we inverse the generation distribution of vehicles. Hence, to simulate this perturbation, we generated for this simulation more vehicles per lane in horizontal direction than in the vertical direction. More precisely, we generated 864 vehicles per hour per lane in the horizontal direction and 432 in the vertical direction. Results in Table III show that the learned policy is better to adapt to changes in traffic flow. However, results are not very good because the learned policy has only been learned in a vertical congestion situation.

System	Avg delay	ST DEV
Pre-timed near-optimal for vertical	77.21	4.17
Learned	62.92	4.50

TABLE III

AVERAGE AND STANDARD DEVIATION FOR VEHICLE DELAY FOR A FOUR INTERSECTIONS WITH MORE HORIZONTAL TRAFFIC (CALCULATED OVER 50 SIMULATIONS)

VII. DISCUSSION AND FUTURE WORK

The work described here illustrates our first efforts towards the optimization of the traffic light controllers on a road network. Through our work in this domain of application of agents, we were able to observe the complexity and some issues related to this problem. Many ideas have come to our mind while working and trying to obtain good control policies. We fully understand why authors of work cited earlier simplified the problem and often noted its complexity. Indeed, many details about vehicle speeds, signal stage times and the definition of the metrics to optimize make learning in this environment a difficult task. To make good decisions, states would need to include many subtle details about the environment that do change and modify the efficiency of an action.

Clearly, an aspect to consider next is to learn in a more general environment, where our controller could face most situations. This would an interesting problem to tackle, as non-stationarity is a major problem in traffic management. On the other hand, the next step would also be to consider multiagent interactions and learning between agent controllers in order to obtain coordination between intersections. This could help distribute the traffic flow when facing unplanned external events.

This work gave us a good idea on how to design fixed controllers for them to perform well. But even then, designing an optimal fixed controller for this problem (which means making the assumption of a stationary traffic flow) has taken us a few trials by adapting the signal stage times. This is where the agent approach shines, as it is more simple as we just let our agents evolve in the environment to get a good resulting policy. Our experimentations showed that our traffic light controller did well against some fixed controllers. Our future work will focus on integrating coordination between intersections and try to design better controllers that can optimize global performance metrics.

VIII. CONCLUSION

We presented a traffic light controller based on intelligents agents that control the shifting of the traffic signal stages at intersections. We showed that our learned policy performs as good enough than pre-timed near-optimal controllers, and better under different traffic loads.

REFERENCES

[1] G. Improta and G. E. Cantarella, "Control system design for an individual signalized junction," *Transportation research. Part B : methodological*, vol. 18B, no. 2, pp. 147–168, 1984.

[2] R. Vincent and C. Young, "Self-optimising traffic signal control using microprocessors - the TRRL 'MOVA' strategy for isolated intersections," *Traffic Engineering and Control*, vol. 27, no. 7, pp. 385–387, 1986.

[3] K. J. Button and D. A. Hensher, Eds., *Handbook of Transport Systems and Traffic Control*. Pergamon, Elsevier, 2001.

[4] D. I. Robertson and R. D. Bretherton, "Optimizing networks of traffic signal in real time – the SCOOT method," *IEEE Transaction on Vehicular Technology*, vol. 40, no. 1, pp. 11–15, 1991.

[5] M. Wiering, "Multi-Agent Reinforcement Learning for Traffic Light Control," in *Seventeeth International Conference on Machine Learning and Applications*, 2000, pp. 1151–1158.

[6] K. Dresner and P. Stone, "Traffic intersections of the future," in *American Association for Artificial Intelligence (AAAI)*, 2006, pp. 1593–1596.

[7] A. L. C. Bazzan, "A distributed approach for coordination of traffic signal agents," *Autonomous Agents and Multi-Agent Systems*, vol. 10, pp. 131–164, 2005.

[8] A. L. C. B. Bruno Castro da Silva, Denise de Oliveria and E. W. Basso, "Adaptive traffic control with reinforcement learning," in *Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2006, pp. 80–86.

[9] T. L. Thorpe and C. W. Anderson, "Traffic light control using sarsa with three state representations," 1996. [Online]. Available: citeseer.ist.psu.edu/thorpe96traffic.html

[10] T. Thorpe, "Vehicle traffic light control using sarsa," 1997. [Online]. Available: citeseer.ist.psu.edu/thorpe97vehicle.html

[11] T. H. Heung, T. K. Ho, and Y. F. Fung, "Coordinated road-junction traffic control by dynamic programming," *IEEE Transaction on Intelligent Transportation Systems*, vol. 6, no. 3, pp. 341–350, September 2005.

[12] M. Wooldridge, *An Introduction to Multiagent Systems*. John Wiley and Sons, 2002, iSBN = 0-471-49691-X.

[13] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. MIT Press, 1998, iSBN = 0-262-19398-1.

[14] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Machine Learning*, vol. 8, no. 3-4, pp. 279–292, 1992.

[15] J. Baxter and P. L. Bartlett, "Infinite-horizon policy-gradient estimation," *Journal of Artifical Intelligence Research*, pp. 319–350, 2001.

[16] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine Learning*, vol. 8, pp. 229–256, 1992.