Reinforcement Learning-based Control of Traffic Lights in Non-stationary Environments: A Case Study in a Microscopic Simulator

Denise de Oliveira [a], Ana L. C. Bazzan [a], Bruno C. da Silva [a], Eduardo W. Basso [a], Luis Nunes [cb], Rosaldo Rossetti [b], Eugénio de Oliveira [b], Roberto da Silva [a], Luis Lamb [a]

[a]Instituto de Informática – UFRGS
C. P. 15064 CEP 91.501-970, P. Alegre, RS, Brazil
[b]NIAD&R – Faculdade de Engenharia da Universidade do Porto
[c]Depto. de Ciências e Tecnologias da Informação – ISCTE, Av. das Forças Armadas, 1649-026, Lisboa, Portugal
{edenise,bazzan,bcs,ewbasso,rdasilva,lamb}@inf.ufrgs.br
eco@fe.up.pt
Luis.Nunes@iscte.pt
r.rossetti@acm.org

Abstract

Coping with dynamic changes in traffic volume has been the object of recent publications. Recently, a method was proposed, which is capable of learning in non-stationary scenarios via an approach to detect context changes. For particular scenarios such as the traffic control one, the performance of that method is better than a greedy strategy, as well as other reinforcement learning approaches, such as Q-learning and Prioritized Sweeping. The goal of the present paper is to assess the feasibility of applying the above mentioned approach in a more realistic scenario, implemented by means of a microscopic traffic simulator. We intend to show that to use of context detection is suitable to deal with noisy scenarios where non-stationarity occurs not only due to the changing volume of vehicles, but also because of the random behavior of drivers in what regards the operational task of driving (e.g. deceleration probability). The results confirm the tendencies already detected in the previous paper, although here the increase in noise makes the learning task much more difficult, and the correct separation of contexts harder.

1 Introduction

Traffic in urban areas is getting increasingly more complex. Since the expansion of the traffic network is no longer a socially attainable solution, the existing control systems have to be used in a more intelligent way in order to increase the traffic throughput and decrease total travel times. The most common way to control traffic is to deploy traffic lights at street junctions. This can solve safety problems, but at the same time causes a decrease of flow and an increase of travel times. In order to minimize the delays, several methods of traffic light control have been proposed.

Classic traffic engineering approaches for controling traffic work well only in traffic networks with very well-defined traffic volume patterns, like for instance morning and afternoon peaks. However, in cities where these patterns are not clear, this approach may not be effective. This is the case in big cities where business centers are no longer located exclusively downtown. For these situations, more flexible approaches are needed.

Classical traffic engineering approaches usually rely on linear programming. This is so because the alternative of using totally decentralized approaches could impose communication bottlenecks for the negotiation, and/or would require a traffic expert to mediate possible conflicts. Thus, more robust approaches than linear programming are not only attractive, but necessary.

Computer science approaches, on the other hand, sometimes make unrealistic assumptions which make them difficult to deploy. This paper seeks to come closer to the reality by *i)* modeling drivers in a more realistic way, and *ii)* using a microscopic traffic simulator less dependent on abstract assumptions, such as those required by simulators based on queues.

The goal of this paper is to extend the approach presented in [10, 11], which uses different flavors of reinforcement learning to control traffic lights in isolated junctions. Specifically, our work intends to assess the feasibility of applying that approach in microscopic simulations, since these are more realistic than the macroscopic approach used before. The use of microscopic simulations allows us to change parameters associated with specific types of drivers, such as their deceleration probability. These type of parameters are important since they are a know source of noise in traffic scenarios. In the previous work, a macroscopic, queue-based modeling was used, and thus many of the fine-grained properties of drivers could not be modeled.

In order to address a finer-grained description of drivers, behavioral models need to possess deliberative components which implement goal-driven behavior. These components are not compatible with the basic Nagel-Schreckenberg model, which is based on a cellular-automata, further detailed in Section 3. A scenario in which drivers are capable of deliberative decision-taking is presented in [2]. However, in this paper a macroscopic simulator is used and hence it is not possible to cope with drivers decelerating, for instance. On the other hand, in [2] drivers are not particles as in [7] and in the present paper. Rather, drivers known their routes and learn how to select them given the traffic state. Traffic state is mostly influenced by traffic lights, which also learn how to select signal plans to cope with the flow of drivers in their areas (co–evolution).

The present paper is organized as follows. In Section 2 we review related approaches to traffic control (Section 2.1) and basic concepts of reinforcement learning and their use in non-stationary environments (Section 2.2). Section 3 introduces some concepts about cellular automata-based microscopic simulation. Section 4 presents the formulation of the problem as one of reinforcement learning while Section 5 discusses the simulations and the results obtained. We present the conclusions in Section 6.

2 Background and Related Work

Signalized intersections are controlled by *signal-timing plans* which are implemented at traffic lights. A signal-timing plan (we use *signal plan* for short) is a unique set of timing parameters comprising the cycle length L (the length of time for the complete sequence of the phase changes), and the split (the division of the cycle length among the various movements or phases). Several plans are normally required for an intersection to deal with changes in traffic volume.

2.1 MAS based approaches to traffic light control

So far, several multi-agent approaches to traffic modelling have been proposed. However, usually the focus of these approaches has been on the management level. On the other hand, our work focuses on a fine-grained level or traffic flow control.

In [1], a MAS based approach is described in which each traffic light is modeled as an agent, each having a set of pre-defined signal plans to coordinate with neighbors. Different signal plans can be selected in order to coordinate in a given traffic direction or during a pre-defined period of the day. This approach uses techniques of evolutionary game theory: self-interested agents receive a reward or a penalty given by the environment. Moreover, each agent possesses only information about its local traffic state. However, payoff matrices (or at least the utilities and preferences of the agents) are required, i.e these figures have to be explicitly formalized by the designer of the system.

In [9], the formation of groups of traffic lights was considered in an application of a technique of distributed constraint optimization, called cooperative mediation. However, in this case the mediation was not decentralized: group mediators communicated their decisions to the mediated agents in their groups and these agents just carried the tasks out. Also, the mediation process could require a long time in highly constrained scenarios, imposing a negative impact in the coordination mechanism. For these reasons, a decentralized and swarm-based model of task allocation was developed in [5], in which the dynamic group formation, without mediation, combined the advantages of those two previous works (decentralization via swarm intelligence and dynamic group formation).

Camponogara and Kraus [3] have studied a simple scenario with only two intersections, using stochastic game-theory and reinforcement learning. Their results with this approach were better than a *best-effort* (greedy) technique, better than a random policy, and also better than Q-learning. Also, in [8] a set of different techniques were applied in order to try to improve the learning ability of the agents in a simple scenario.

2.2 Reinforcement Learning

Usually, Reinforcement Learning (RL) problems are modeled as Markov Decision Processes (MDPs). These are described by a set of states, $\mathcal{S}$, a set of actions, $\mathcal{A}$, a reward function $R(s,a) \rightarrow \Re$ and a probabilistic state transition function $T(s,a,s') \rightarrow [0,1]$. An experience tuple $\langle s,a,s',r \rangle$ denotes the fact that the agent was in state s, performed action a and ended up in s' with reward r.

2.2.1 Q-Learning

Reinforcement learning methods can be divided into two categories: model-free and model-based. Model-based methods assume that the transition function T and the reward function R are available. Model-free systems, such as Q-learning, on the other hand, do not require that the agent have access to information about how the environment works. Q-Learning works by estimating good state–action values, the Q-values, which are a numerical estimator of quality for a given pair of state and action.

Q-Learning algorithm approximates the Q-values $Q(s,a)$ as the agent acts in a given environment. The update rule for each experience tuple $\langle s,a,s',r \rangle$ is:

$$Q(s,a) = Q(s,a) + \alpha \; (r + \gamma max_{a'} \, Q(s',a') - Q(s,a))$$

where α is the learning rate and γ is the discount for future rewards. When the Q-values are nearly converged to their optimal values, it is appropriate for the agent to act greedily, that is, to always choose the action with the highest Q-value for the current state.

2.2.2 Prioritized Sweeping

Prioritized Sweeping (PS) is somewhat similar to Q-Learning, except for the fact that it continuously estimates a single model of the environment. Also, it updates more than one state value per iteration and makes direct use of state values, instead of Q-values. The states whose values should be updated after each iteration are determined by a priority queue, which stores the states *priorities*, initially set to zero. Also, each state remembers its predecessors, defined as all states with a non-zero transition probability to it under some action. At each iteration, new estimates $\hat{T}$ and $\hat{R}$ of the dynamics are made. The usual manner to update $\hat{T}$ is to calculate the maximum likelihood probability. Instead of storing the transitions probabilities in the form $T(s,a,s')$, PS stores the number of times the transition (s,a,s') occurred and the total number of times the situation (s,a) has been reached. This way, it is easy to update these parameters and to compute $\hat{T}$ as $\frac{|\,(s,a,s')\,|}{|\,(s,a)\,|}$.

2.2.3 RL in non-stationary environments

When dealing with non-stationary environments, both the model-free and the model-based RL approaches need to continuously relearn everything from scratch, since the policy which was calculated for a given environment is no longer valid after a change in dynamics. This causes a performance drop during the readjustment phase, and also forces the algorithm to relearn policies even for environment dynamics which have been previously experienced.

In order to cope with non-stationary environments, alternative RL methods have been proposed, e.g. the mechanisms proposed by Choi and colleagues [4] and Doya and colleagues [6]. Unfortunately, their approaches require a fixed number of models, and thus implicitly assume that the approximate number of different environment dynamics is known *a priori*. Since this assumption is not always realistic, our method tries to overcome it by incrementally building new models.

3 Microscopic Simulation

There are two main approaches to the simulation of traffic: macroscopic and microscopic. The latter allows the description of each road user as detailed as desired, given computational restrictions, thus permitting the modeling several types of drivers' behaviors. Multi-agent simulation is a promising technique for microscopic traffic models as the drivers' behavior can be described in a way that incorporates complex and individual decision-making.

In general, microscopic traffic flow models describe the act of driving on a road, that is, the perception and reaction of a driver on a short time-scale. In these models, the drivers are the basic entities, and their behaviors are described according to several different types of mathematical formulations, (e.g. continuous models such as car-following). Other models are not continuous, but instead use cellular automata techniques. In particular, we use the Nagel-Schreckenberg model [7] due to its simplicity. The Nagel-Schreckenberg cellular automaton-model represents a *minimal model* in the sense that it is capable to reproduce basic several features of real traffic using only very few behavioral rules.

In the following, the Nagel-Schreckenberg model for single-lane traffic is briefly reviewed. The road is considered to be subdivided in cells with a length varying around 7.5 or 5 meters (for highways or urban traffic respectively). Each cell is either empty or occupied by exactly one vehicle, which has an integer speed $v_i \in \{0, \ldots, v_{max}\}$, and a maximum speed v_{max}. The dynamics of vehicle motion is described by the following rules (*parallel dynamics*):

R1 Acceleration: $v_i \leftarrow min(v_i + 1, v_max)$,

R2 Deceleration: in order to avoid accidents, $v'_i \leftarrow \min(v_i, gap)$,

R3 Randomizing: with a certain probability p
do $v''_i \leftarrow \max(v'_i - 1, 0)$,

R4 Movement: $x_i \leftarrow x_i + v''_i$.

The variable *gap* denotes the number of empty cells in front of the vehicle at cell i. A time-step corresponds to $\Delta t \approx 1$ sec, the typical time a driver needs to react. Every driver described by the Nagel-Schreckenberg model can be seen as a reactive agent: autonomous, situated in a discrete environment, and having (potentially individual) characteristics such as its maximum speed v_{max} and deceleration probability p. During the process of driving, it perceives its distance to the predecessor (*gap*), its own current speed v, etc. These information are processed using rules **R1-R3**, and changes in the environment are made using **R4**. The first rule describes one goal of the agent (to move according to the maximum speed possible, v_{max}); its other goal is to drive safely i.e. not to collide with its predecessor (**R2**). These first two rules describe deterministic behavior, which implies that stationary state of the system is determined only by the initial conditions. However, real drivers do not react in this optimal way. Instead, sometimes they vary their driving behavior without any obvious reasons. This uncertainty in behavior is reflected by the *braking noise p* (**R3**). Braking noise mimics the complex interactions with the other agents, such overreaction in braking and delayed reaction while accelerating. Finally, to carry the last rule out means that the agent will act on the environment and move according to its current speed.

In the next section, we discuss the use of the microscopic modeling in a scenario in which the traffic lights learn to cope with the different traffic patterns. All experiments were made using the ITSUMO tool [12] which implements the Nagel-Schreckenberg model.

4 Using RL-CD for Traffic Lights Control

Traffic optimization is a hard problem because it deals with highly dynamic and non-stationary flow patterns. Thus, it is important to state our assumptions regarding the type of dynamics we expect to happen. Specifically, the class of non-stationary environments that we deal with is similar to the one studied by Hidden-Mode MDPs researchers [4]. We assume that the following properties hold: **1)** environmental changes are confined to a small number of *contexts*, which are stationary environments with distinct dynamics; **2)** the current context cannot be directly observed, but can be estimated according to the types of transitions and rewards observed; **3)** environmental context changes are independent of the agent's actions; and **4)** context changes are relatively infrequent.

In a traffic scenario, these assumptions mean that flow patterns are non-stationary but they can be nearly divided in stationary dynamics. We do not assume, however, that this division must be known *a priori*. In fact, one of the interesting aspects of our method is exactly its capability of automatically partitioning the environment dynamics into relevant partial models. Moreover, we assume that the current flow pattern is not given or directly perceivable, but can be estimated by observing attributes such as the queue of vehicles, street densities, etc.

Our approach relies on RL methods because they provide online learning capabilities and do not rely on offline analysis. In the next sections we review the RL method called RL-CD or Reinforcement Learning with Context Detection [11]. The RL-CD mechanism assumes that the use of multiple partial models of the environment is a good approach for dealing with non-stationary scenarios such as traffic control. The use of multiple models makes the learning system capable of partitioning the knowledge in a way that each model automatically assumes for itself the responsibility for "understanding" one kind of flow pattern. To each model, we assign an optimal policy, which is a mapping from traffic conditions to signal plans, and a trace of prediction error of transitions and rewards, responsible for estimating the quality of a given partial model. Moreover, the creation of new models is controlled by a continuous evaluation of the prediction errors generated by each partial model. In the following subsections we first describe how to learn contexts (i.e, estimate models for traffic patterns), and then we show how to detect and switch to the most adequate model given a sequence of observations.

4.1 Learning traffic patterns

Henceforth we use the terms *traffic pattern* and *context* interchangeably. A context consists of a nearly stable set of traffic flow characteristics. In terms of RL, a context consists of a given environment dynamics, which is experienced by the agent as a class of transitions and rewards. The mechanism for detecting context changes relies on a set of partial models for predicting the environment dynamics. A partial model m contains a function $\hat{T}_m$, which estimates transition probabilities, and also a function $\hat{R}_m$, which estimates the rewards to be received.

For each partial model, classic model-based RL methods such as Prioritized Sweeping and Dyna [13] may be used to compute a locally optimal policy. The policy of a partial model is described by the function $\pi_m(s)$ and it is said to be locally optimal because it describes the optimal actions for each state in a specific context. For example, if the dynamics of a non-stationary environment Θ can be described by m_1, then π_{m_1} will be the associated optimal policy. If the non-stationarity of Θ makes itself noticeable by making π_{m_1} suboptimal, then the system creates a new model, m_2, which would predict with a higher degree of confidence the transitions of the newly arrived context. Associated with m_2, a locally optimal policy π_{m_2} would be used to estimate the best actions in m_2. Whenever possible, the system reuses existing models instead of creating new ones.

Given an experience tuple $\varphi \equiv \langle s, a, s', r \rangle$, we update the current partial model m by adjusting its model of transition and rewards by $\Delta^{\hat{T}}_{m,\varphi}$ and $\Delta^{\hat{R}}_{m,\varphi}$, respectively. These adjustments are computed as follows:

$$\Delta^{\hat{T}}_{m,\varphi}(\kappa) = \tfrac{1}{\mathcal{N}_m(s,a)+1}\left(\tau^{s'}_{\kappa} - \hat{T}_m(s,a,\kappa)\right) \quad \forall \kappa \in \mathcal{S}$$

$$\Delta^{\hat{R}}_{m,\varphi} = \tfrac{1}{\mathcal{N}_m(s,a)+1}\left(r - \hat{R}_m(s,a)\right)$$

such that τ is the Kronecker Delta:

$$\tau^{s'}_{\kappa} = \begin{cases} 1, & \kappa = s' \\ 0, & \kappa \neq s' \end{cases}$$

The effect of τ is to update the transition probability $T(s,a,s')$ towards 1 and all other transitions $T(s,a,\kappa)$, for all $\kappa \in \mathcal{S}$, towards zero. The quantity $\mathcal{N}_m(s,a)$ reflects the number of times, in model m, action a was executed in state s. We compute $\mathcal{N}_m$ considering only a truncated (finite) memory of past M experiences:

$$\mathcal{N}_m(s,a) = min\Big(\mathcal{N}_m(s,a) + 1,\, M\Big) \tag{1}$$

A truncated value of $\mathcal{N}$ acts like a learning coefficient for $\hat{T}_m$ and $\hat{R}_m$, causing transitions to be updated faster in the initial observations and slower as the agent experiments more. Having the values for $\Delta^{\hat{T}}_{m,\varphi}$ and $\Delta^{\hat{R}}_{m,\varphi}$, we update the transition probabilities:

$$\hat{T}_m(s,a,\kappa) = \hat{T}_m(s,a,\kappa) + \Delta^{\hat{T}}_{m,\varphi}(\kappa), \quad \forall \kappa \in \mathcal{S} \tag{2}$$

and also the model of expected rewards:

$$\hat{R}_m(s,a) = \hat{R}_m(s,a) + \Delta^{\hat{R}}_{m,\varphi} \tag{3}$$

4.2 Detecting changes in traffic patterns

In order to detect changes in the traffic patterns (contexts), the system must be able to evaluate how well the current partial model can predict the environment. Thus, an error signal is computed for each partial model. The *instantaneous error* is proportional to a *confidence value*, which reflects the number of times the agent tried an action in a state. Given a model m and an experience tuple $\varphi = \langle s, a, s', r \rangle$, we calculate the instantaneous error $e_{m,\varphi}$ and the confidence $c_m(s,a)$ as follows:

$$c_m(s,a) = \left(\frac{\mathcal{N}_m(s,a)}{M}\right)^2 \tag{4}$$

$$e_{m,\varphi} = c_m(s,a)\Big(\Omega(\Delta^{\hat{R}}_{m,\varphi})^2 + (1-\Omega)\sum_{\kappa \in \mathcal{S}} \Delta^{\hat{T}}_{m,\varphi}(\kappa)^2\Big) \tag{5}$$

where Ω specifies the relative importance of the reward and transition prediction errors for the assessment of the model's quality. Once the instantaneous error has been computed, the trace of prediction error E_m for each partial model is updated:

$$E_m = E_m + \rho\Big(e_{m,\varphi} - E_m\Big) \tag{6}$$

where ρ is the adjustment coefficient for the error.

The error E_m is updated after each iteration for every partial model m, but only the active model is corrected according to equations 2 and 3. A plasticity threshold λ is used to specify until when a partial model should be adjusted. When E_m becomes higher than λ, the predictions made by the model are considered sufficiently different from the real observations. In this case, a context change is detected and the model with lowest error is activated. A new partial model is created when there are no models with trace error smaller than the plasticity. The mechanism starts with only one model and then incrementally creates new partial models as they become necessary. In other words, whenever the model of predictions is considered sufficiently inadequate, we assume that the dynamics of the environment have changed, and then we either select among some other good model, or create a new one from scratch.

The parameter λ must be adjusted according to how non-stationary the environment seems to be. Among the factors that influence this we can name *i)* how coarse the state discretization is; *ii)* the number of Markov transitions whose probabilities change when the context is altered; *iii)* the quantity of aliased or partially-observable states. After λ is adjusted and exploration takes place, during learning, we expect a finite number of models to be created. The total number of models should stabilise, and these models are expected to correctly describe the stationary dynamics that compose the overall non-stationary scenario. In our experiments, we have seen no relation between λ and the RL learning rate α, since the first measures the maximum acceptable prediction error (related to the environment model), and the second measures how fast the Q-values are updated (related to the policy).

Pseudo-code for RL-CD is presented in Algorithm 1.

The *newmodel*() routine is used to create a new partial model and initializes all estimates and variables to zero, except T_m, initialized with equally probable transitions. The values of parameters

Algorithm 1 RL-CD algorithm

```
Let $m_{cur}$ be the currently active partial model.
Let $\mathcal{M}$ be the set of all available models.
 1: $m_{cur} \leftarrow newmodel()$
 2: $\mathcal{M} \leftarrow \{m_{cur}\}$
 3: $s \leftarrow s_0$, where $s_0$ is any starting state
 4: loop
 5:   Let $a$ be the action chosen by PS for the model $m_{cur}$
 6:   Observe next state $s'$ and reward $r$
 7:   Update $E_m$, for all $m$, according to equation 6
 8:   $m_{cur} \leftarrow \arg\min_m (E_m)$
 9:   if $E_{m_{cur}} > \lambda$ then
10:     $m_{cur} \leftarrow newmodel()$
11:     $\mathcal{M} \leftarrow \mathcal{M} \cup \{m_{cur}\}$
12:   end if
13:   Update $\hat{T}_{m_{cur}}$ and $\hat{R}_{m_{cur}}$ (equations 2 and 3)
14:   $\mathcal{N}_m(s, a) \leftarrow \min(\mathcal{N}_m(s, a) + 1, M)$
15:   $s \leftarrow s'$
16: end loop
```

M, ρ, Ω and λ must be tuned according to the problem. Small values of ρ are appropriate for noisy environments; higher values of M define systems which require more experiences in order to gain confidence regarding its predictions; in general applications, Ω might be set to 0.5; the plasticity λ should be set to higher values according to the need for learning relevant (non-noisy) but rare transitions.

5 Scenario and Experiments

5.1 Scenario

Our scenario consists of a traffic network which is a 3x3 Manhattan-like grid, with a traffic light in each junction. Figure 1 depicts this network, where the 9 nodes correspond to traffic lights and the 24 edges are directed (one-way) links, each having a specific maximum allowed velocity. In Figure 1, the links depicted in black, dotted, and gray have their maximum allowed velocities set to 54Km/h, 36km/h, and 18Km/h respectively. These velocities correspond to 3, 2, and 1 cell per time-step.

Each link has capacity for 50 vehicles. Vehicles are inserted by *sources* and removed by *sinks*, depicted in the figure as diamonds and squares, respectively, in Figure 1. In order to model the non-stationarity of the traffic behavior, our scenario assumes three different traffic patterns (contexts). Each consists of a different vehicle insertion distribution. These three contexts are:

- *Low*: low insertion rate in the both North and East sources, allowing the traffic network to perform relatively well even if the policies are not optimal (i.e., the network is undersaturated);
- *Vertical*: high insertion rate in the North sources (S0, S1, and S2), and average insertion rate in the East (S3, S4, and S5);
- *Horizontal*: high insertion rate in the East sources (S3, S4, and S5), and average insertion rate in the East (S0, S1, and S2).

Vehicles do not change directions during the simulation, and upon arriving at sinks they are immediately removed.

We modeled the scenario in a way that each traffic light is controlled by exactly one agent, each agent making only local decisions. Even though decisions are local, we assess how well the mechanism is performing by measuring global performance values. By using reinforcement learning to optimize isolated junctions, we implement decentralized controllers and avoid expensive offline processing.

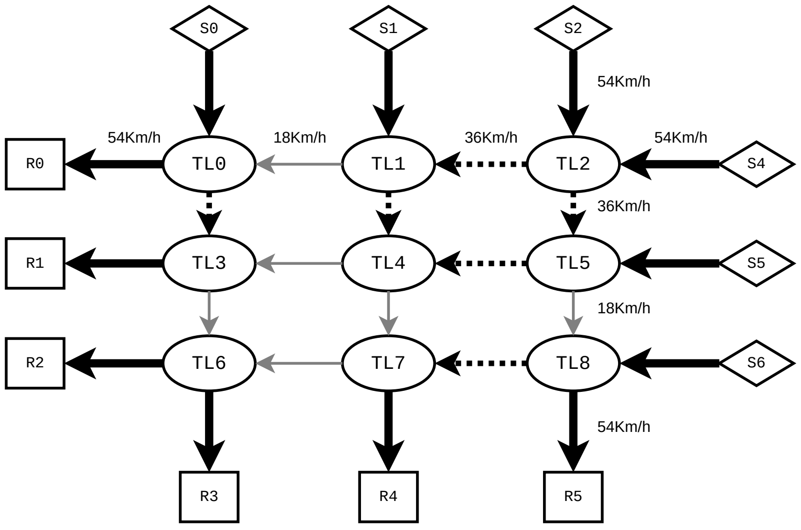

Figure 1: A Network with Nine Intersections.

As a general measure of performance or effectiveness of traffic control systems, usually one seeks to optimize a weighted combination of number of stopped vehicles and total travel time. Here we measure the total number of stopped vehicles, since this is an attribute which can be easily delivered by real inductive loop detectors.

At each time, we discretize the number of cars in a link in a way that its state can be either *empty*, *regular* or *full*, based on its occupation. The state of an agent is given by the states of the links arriving in its corresponding traffic light. Since there are two one-way links arriving at each traffic light (one from north and one from east), there are 9 possible states *for each agent*. The reward for each agent is given locally by the squared sum of the incoming links' queues. Performance, however, is evaluated for the whole traffic network by summing the queue length of all links, including external queues (queues containing vehicles trying to enter the scenario).

We have already discussed that traffic lights usually need a set of signal plans in order to deal with the different traffic conditions and/or time of the day. We consider here three plans, each with two phases: one allowing green time to direction north-south (NS), and other to direction east-west (EW). Each one of the three signal plans uses different green times for phases: signal plan **1** gives equal green times for both phases; signal plan **2** gives priority to the vertical direction; and signal plan **3** gives priority to the horizontal direction. All signal plans have cycle time of 60 seconds and phases of either 42, 30 or 18 seconds (70% of cycle time for the preferential direction, 50% of cycle time and 25% of cycle time for non-preferential direction). The signal plan with equal phase times gives 30 seconds for each direction (50% of the cycle time); the signal plan which prioritizes the vertical direction gives 42 seconds to the phase NS and 18 seconds to the phase EW; and the signal plan which prioritizes the horizontal direction gives 42 seconds to the phase EW and 18 seconds to the phase NS.

In our simulation, one time-step consists of two entire cycles of signal plan, that is, 120 seconds of simulated traffic. The agent's action consists of selecting one of the three available signal plans at each simulation step.

5.2 Experiments

In the following experiments we compare our method against a greedy policy and against classic model-free and a model-based reinforcement learning algorithms.

We change the context (traffic pattern) every 100 time-steps, which corresponds to 12000 seconds. The probability of a vehicle being inserted, at each second, during each traffic pattern by the sources is given in Table 1. Moreover, all comparisons of control methods' performance make use of the metric described in section 5.1, that is, the total number of stopped vehicles in all links, including external queues, during the simulation time. The use of this metric implies that the lower

	Sources	
Traffic pattern	S0, S2, S3	S4, S5, S6
Low	0.05	0.05
Vertical	0.20	0.05
Horizontal	0.05	0.20

Table 1: Insertion Rates

the value in the graph, the better the performance.

We have first implemented a greedy policy, for the sake of comparison. The greedy solution is a standard decentralized solution for traffic-responsive networks in which there is no type of coordination. Each agent takes decisions based solely on the state of its input links, and then selects the signal plan which gives priority to the direction with higher occupation. If the state of both links is the same, then the greedy agent selects the signal plan with equal time distribution.

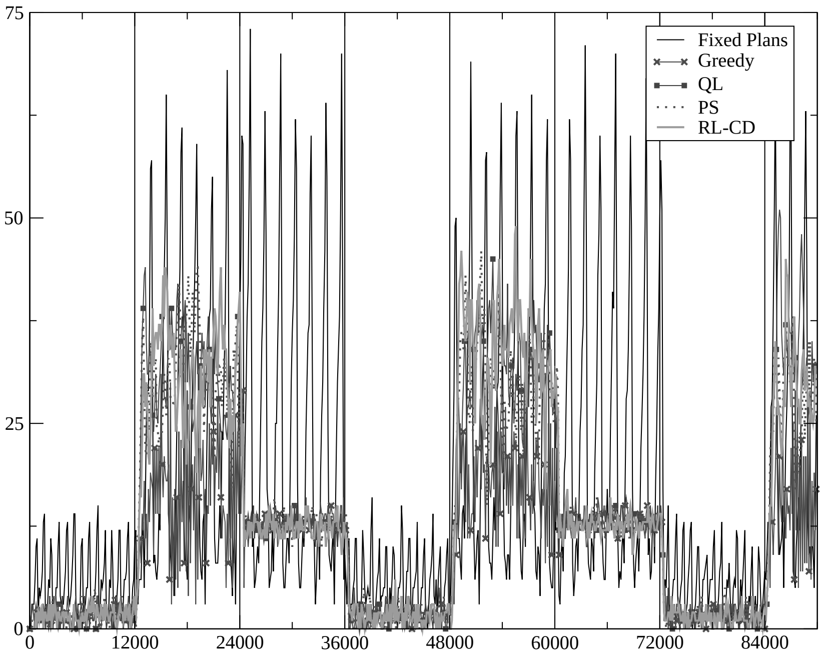

Figure 2: Number of stopped vehicles (scenario with deceleration probability set to zero)

Figure 2 shows the comparison between our method, fixed signal plans, greedy selection, and two standard RL methods, namely Q-Learning and Prioritized Sweeping, in a scenario where the vehicles have no probability of deceleration (see rule **R3** in Section 3). We can see on this graph that the performance of all learning methods is very similar to the greedy selection mechanism. Using no learning, i.e. using fix plans, is of course not efficient. This is so because fix plans do not respond to the change in number of vehicles as it gives the same priority to both directions.

That figure also shows that, in an ideal scenario, with no noise (no deceleration), our mechanism has a performance similar to the greedy and to other learning methods. Although in this case the greedy strategy is the best, and therefore no learning occurs, we can imagine that better solutions exist in which the system learns to identify and to deal with the different types of traffic patterns.

Next, in Figure 3 we show the performance of all methods in a scenario in which vehicles have a probability of deceleration equal to 0.1. In this case, the braking noise makes the greedy method and Q-Learning have a lower performance compared to our method and PS. Q-learning performs bad because it learns to cope with equal flow during the first context and then has troubles relearning the next contexts.

The results in the other two scenarios (scenarios with deceleration 0.2 and 0.3) are shown

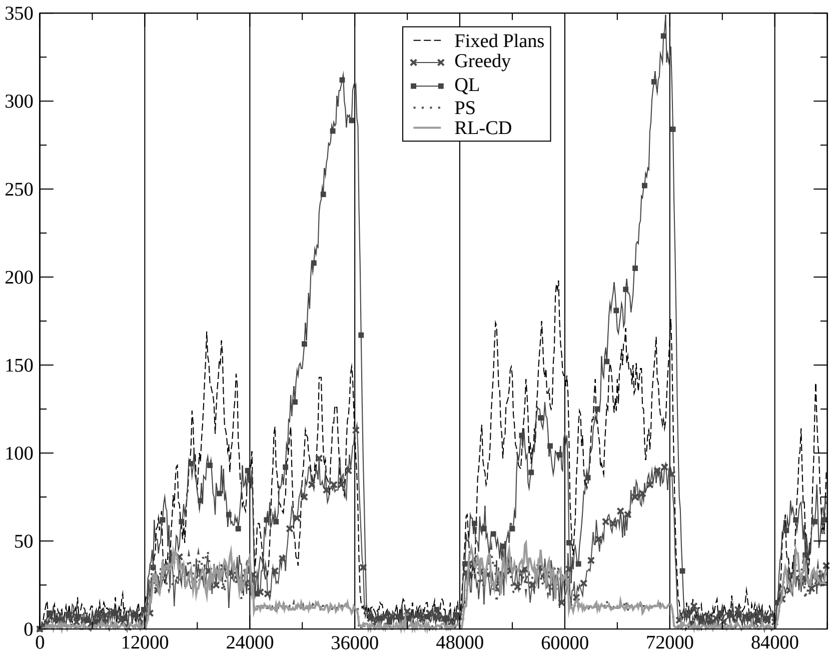

Figure 3: Number of stopped vehicles (scenario with deceleration probability set to 0.1)

in Figure 4 and Figure 5 respectively. As we can observe, the deceleration p causes the traffic lights to always observe the same state (in this case, *high occupation in all input lanes*), which means that none of the learning methods is able to cope with the over-saturated network. For this same reason, RL-CD is not able to build interesting models of the relevant attributes of the dynamics. However, since Q-Learning is model-free, it is less prone to wrong bias caused by the non-stationarity. Prioritized Sweeping, on the other hand, tries to build a single model for the environment and ends up with a model which mixes properties of different traffic patterns. For this reason, it can at most calculate a policy which is a compromise for several different (and sometimes opposite) traffic patterns. The fixed plan and the learning mechanisms have the same performance, indicating that the learning mechanisms are using only the fixed plan.

6 Conclusions

Centralized approaches to traffic signal control cannot cope with the increasing complexity of urban traffic networks. A trend towards decentralized control was already pointed out by traffic experts in the 1980's and traffic responsive systems for control of traffic lights have been implemented.

Here we have used reinforcement learning methods capable of dealing with non-stationary traffic patterns in a *microscopic simulator.* This kind of simulator allows us to test several sources of non-stationarity. In this paper, we considered specifically the deceleration probability, for it is an important characteristic of drivers not present in macroscopic models of simulation. The results show that in face of higher degree of non-stationarity the learning mechanisms have more difficulty in recognizing and discriminating the states than when the macroscopic model was used (for example, in [10]). Moreover, we have presented empirical results which show that RL-CD is more efficient than classical Q-Learning and a greedy method in a non-stationary noisy scenario, provided that real traffic state is not aliased by coarse discretization. In other words, the use of reinforcement learning techniques, such as RL-CD, Q-Learning and Prioritized Sweeping, requires that either the network is not over-saturated at all times, or that states are discretized in a finer-grained manner.

The conclusion regarding the bad performance of Q-learning can be generalized to any scenario involving learning in traffic lights, and was in fact expected. This is so because Q-learning is based on an experimentation over the whole table of state-action pairs. However, it is clear that some of

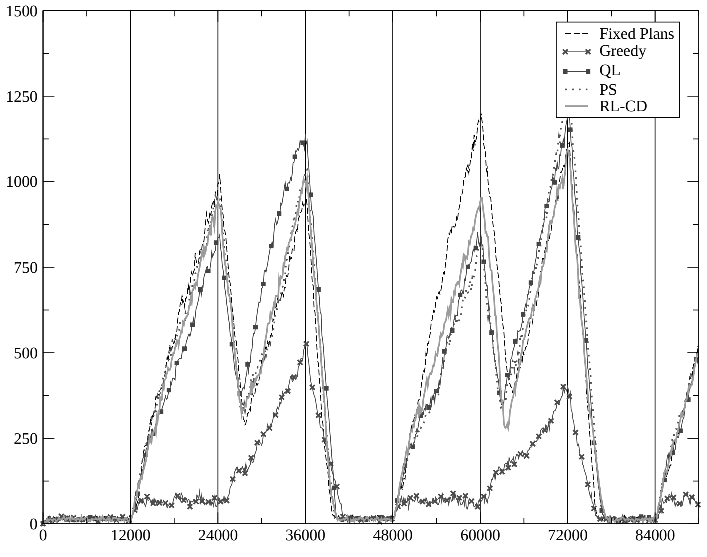

Figure 4: Number of stopped vehicles (scenario with deceleration probability set to 0.2)

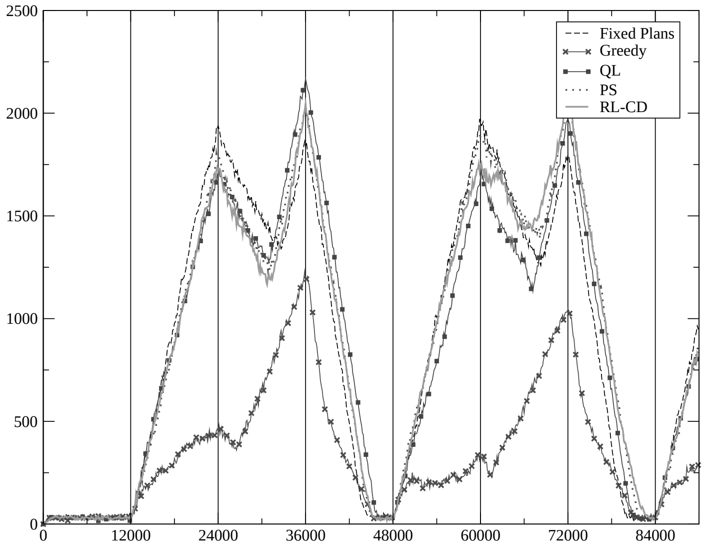

Figure 5: Number of stopped vehicles (scenario with deceleration probability set to 0.3)

these pairs make no sense and should not have been tried. For example, if $TL4$ in Figure 1 has state *empty* in the vertical direction and *full* in the horizontal direction, then Q-learning should not even try to run the action *signal plan 2* (which priorizes the vertical direction), but it does for the sake of exploration, thus increasing the congestion level in the horizontal link. This means that one must be cautious when using methods which require real-time learning (such as RL) in online scenarios like traffic control.

We are extending the ITSUMO tool to make possible the simulation of goal-driven drivers in order to test scenarios with co-learning of traffic lights and drivers in a microscopic simulation model.

Acknowledgments

The joint project “Caracterização de Estratégias de Controle em Sistemas Multiagentes Heterogêneos” is supported by the bilateral agreement between Brazil and Portugal (CAPES-GRICES). Ana Bazzan partially supported by the Alexander v. Humboldt Foundation and by CNPq; Bruno C. da Silva is supported by CNPq, and Denise de Oliveira is supported by CAPES.

References

[1] Bazzan, A. L. C. A distributed approach for coordination of traffic signal agents. *Autonomous Agents and Multiagent Systems 10*, 1 (March 2005), 131–164.

[2] Bazzan, A. L. C., de Oliveira, D., Klügl, F., and Nagel, K. Effects of co-evolution in a complex traffic network. (submitted).

[3] Camponogara, E., and Jr., W. K. Distributed learning agents in urban traffic control. In *EPIA* (2003), F. Moura-Pires and S. Abreu, Eds., pp. 324–335.

[4] Choi, S. P. M., Yeung, D.-Y., and Zhang, N. L. Hidden-mode markov decision processes for nonstationary sequential decision making. In *Sequence Learning - Paradigms, Algorithms, and Applications* (London, UK, 2001), Springer-Verlag, pp. 264–287.

[5] de Oliveira, D., and Bazzan, A. L. C. Traffic lights control with adaptive group formation based on swarm intelligencetraffic lights control with adaptive group formation based on swarm intelligence. In *Proceedings of the 5th International Workshop on Ant Colony Optimization and Swarm Intelligence, ANTS 2006* (Brussels, September 2006), M. Dorigo, L. M. Gambardella, M. Birattari, A. Martinoli, R. Poli, and T. Stuetzle, Eds., Lecture Notes in Computer Science, Springer, pp. 520–521.

[6] Doya, K., Samejima, K., Katagiri, K., and Kawato, M. Multiple model-based reinforcement learning. *Neural Computation 14*, 6 (2002), 1347–1369.

[7] Nagel, K., and Schreckenberg, M. A cellular automaton model for freeway traffic. *Journal de Physique I 2* (1992), 2221.

[8] Nunes, L., and Oliveira, E. C. Learning from multiple sources. In *Proceedings of the 3rd International Joint Conference on Autonomous Agents and Multi Agent Systems, AAMAS* (New York, USA, July 2004), vol. 3, New York, IEEE Computer Society, pp. 1106–1113.

[9] Oliveira, D., Bazzan, A. L. C., and Lesser, V. Using cooperative mediation to coordinate traffic lights: a case study. In *Proceedings of the 4th International Joint Conference on Autonomous Agents and Multi Agent Systems (AAMAS)* (July 2005), New York, IEEE Computer Society, pp. 463–470.

[10] Silva, B. C., Oliveira, D. d., Bazzan, A. L. C., and Basso, E. W. Adaptive traffic control with reinforcement learning. In *Proceedings of the 4th Workshop on Agents in Traffic and Transportation (AAMAS 2006)* (May 2006), A. L. C. Bazzan, B. C. Draa, and F. Kl, Eds., pp. 80–86.

[11] Silva, B. C. d., Basso, E. W., Bazzan, A. L. C., and Engel, P. M. Dealing with non-stationary environments using context detection. In *Proceedings of the 23rd International Conference on Machine Learning (ICML 2006)* (June 2006), W. W. Cohen and A. Moore, Eds., ACM Press, pp. 217–224.

[12] Silva, B. C. d., Junges, R., Oliveira, D. d., and Bazzan, A. L. C. ITSUMO: an intelligent transportation system for urban mobility. In *Proceedings of the 5th International Joint Conference On Autonomous Agents And Multiagent Systems (AAMAS 2006) - Demonstration Track* (May 2006), P. Stone and G. Weiss, Eds., ACM Press, pp. 1471–1472.

[13] Sutton, R. S. Planning by incremental dynamic programming. In *Proceedings of the Eighth International Workshop on Machine Learning* (1991), Morgan Kaufmann, pp. 353–357.