Multiagent Reinforcement Learning for Traffic Signal Control: a k-Nearest Neighbors Based Approach

Vicente N. de Almeida[1], Ana L. C . Bazzan[1] and Monireh Abdoos[2]

[1]*Computer Science, Universidade Federal do Rio Grande do Sul (UFRGS), Porto Alegre, RS, Brazil*
[2]*Faculty of Computer Science and Engineering, Shahid Beheshti University, Tehran, Iran*

Abstract

The increasing demand for mobility in our society poses various challenges to traffic engineering, computer science in general, and artificial intelligence in particular. As it is often the case, it is not possible to increase the capacity of road networks, therefore a more efficient use of the available transportation infrastructure is required. This relates closely to multiagent systems and to multiagent reinforcement learning, as many problems in traffic management and control are inherently distributed. However, one of the main challenges of this domain is that the state space is large and continuous, which makes it difficult to properly discretize states and also causes many RL algorithms to converge more slowly. To address these issues, a multiagent system with agents learning independently via a temporal difference learning algorithm based on k-nearest neighbors is presented as an option to control traffic signals in real-time. Our results show that the proposed method is both effective (reduces the average waiting time of vehicles in a traffic network) and efficient (the learning task is significantly accelerated), when compared to a baseline reported in the literature.

Keywords

multiagent reinforcement learning, k-nearest neighbors, traffic signal control

1. Introduction

Traffic congestion is a phenomenon caused by too many vehicles trying to use the same infrastructure at the same time. The consequences are well-known: air pollution, decrease in speed, delays, opportunity costs, etc. The increase in transportation demand can be met by providing additional capacity. However, this might not be economically or socially attainable or feasible. Thus, optimizing the use of the existing infrastructure is key. One way to accomplish this is by control techniques, notably the adaptive control of traffic signal controllers. A major challenge of optimizing such a controller is that the problem is very constrained, since minimum and maximum green times need to be observed, and the control policy needs to be fair to all traffic directions, even the side ones (in order to deal with starvation).

Several approaches to adaptive control exist (see, e.g., [1]). However, those more sophisticated ones can be applied only to a handful of intersections. Therefore, approaches based

ATT'22: Workshop Agents in Traffic and Transportation, July 25, 2022, Vienna, Austria
vnalmeida@inf.ufrgs.br (V. N. d. Almeida); bazzan@inf.ufrgs.br (A. L. C. . Bazzan); m_abdoos@sbu.ac.ir (M. Abdoos)
http://www.inf.ufrgs.br/~bazzan (A. L. C. . Bazzan)
0000-0002-2803-9607 (A. L. C. . Bazzan)
© 2022 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).
CEUR Workshop Proceedings (CEUR-WS.org)

CEUR-WS.org/Vol-3173/3.pdf

on reinforcement learning (RL) are gaining popularity. In these approaches, learning agents are normally in charge of controlling the signals at a single intersection, in a distributed and decentralized way. We remark that for centralized approaches, where there is a single controller in charge of computing optimal actions for the whole set of intersections, deep learning is more popular. However, due to robustness issues (central point of failure, communication failures), it is desirable to avoid centralized solutions. Besides, centralized approaches assume a central entity in charge of the control, which needs to collect all information from all intersections, and needs to determine an action for each controller at the intersections, thus violating the autonomy of the individual controllers. That being the case, in this paper we deal with multiple agents (signal controllers) learning in a distributed way.

One important aspect of such learning task is that traffic signal control is a highly non-local phenomenon, i.e., it is affected by actions of other agents. This is what makes multiagent RL much more challenging than single agent RL. Another challenge is the fact that the state space is very large (thus RL algorithms like Q-learning converge more slowly) and continuous (appropriately discretizing continuous states is a difficult problem).

That being said, alternatives to tackle these matters, like function approximation, come with some drawbacks. They make the learning harder to comprehend, and generally require that the agents gather a large amount of data to be able to successfully generalize and apply their collected experiences.

To tackle large continuous state spaces in an efficient way, this paper proposes the use of a method that avoids both the need for state discretization and the usage of function approximation, by using a temporal difference learning algorithm based on k-nearest neighbors (k-NN) [2, 3]. This algorithm estimates Q-values by calculating a weighted average of the Q-values of the k closest previously visited states. It is applied to traffic signal control using a multiagent approach, in which each traffic controller (one per intersection) learns independently.

To our best knowledge, a temporal difference learning approach based on k-nearest neighbors has not been used for traffic signal control. Also, in our experiments, we deal with changes in traffic flows, which has rarely been discussed in the traffic signal control with RL literature.

The reader can find details of the proposed method and of how it is applied to traffic signal control in Section 4. In Section 3 we present the related literature, and in Section 2 we discuss the underlying concepts such as RL and traffic signal control. Our results are presented and discussed in Section 5, followed by a conclusion and a discussion on future research lines.

2. Background

This section briefly presents underlying concepts on RL and on traffic signal control.

2.1. Reinforcement Learning

In RL, an agent learns how to act in an environment interacting and receiving a feedback signal (reward) that measures how its action has affected the environment. The agent does not a priori know how its actions affect the environment, hence it has to learn this by trial and error (in an exploration phase). However, the agent should not only explore; in order to maximize the rewards of its action, it also has to exploit the gained knowledge. Thus, there must be an

exploration-exploitation strategy that is to be followed by the agent. One of these strategies is ε-greedy, where an action is randomly chosen (exploration) with a probability ε, or, with probability 1-ε, the best known action is chosen, i.e., the one with the highest Q-value so far (exploitation). In the exploitation phase, at each interaction, it is assumed that the agent has sensors to determine its current state and can then decide on an action. The reward is then used to update its policy, i.e., a mapping from states to actions. This policy can be generated or computed in several ways.

For the sake of the present discussion, we concentrate on a model-free, off-policy algorithm called Q-learning [4], which estimates so-called Q-values using a table to store the experienced values of performing a given action when in a given state. Hence Q-learning is a tabular method, where the state space and the action space need to be discretized.

In RL, the learning task is usually formulated as a Markov decision process (MDP), composed by the sets of states and actions, a transition function, and a reward function. Since the transition and the reward functions are unknown to the agent, its task is exactly to learn them, or at least a model for them.

The value of a state s_t and action a_t at time t is updated based on Eq. (1), where $\alpha \in [0, 1]$ is the learning rate, $\gamma \in [0, 1]$ is the discount factor, s_{t+1} is the next state and r_t is the reward received when the agent moves from s_t to s_{t+1} after selecting action a_t in state s_t.

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha(r_t + \gamma \max_a(Q(s_{t+1}, a)) - Q(s_t, a_t)) \tag{1}$$

When there are multiple agents interacting in a common environment, the RL task (thus, multiagent RL) is inherently more complex than that regarding single-agent RL, because agents' actions are highly coupled and agents are trying to adapt to other agents that are also learning. Besides, several convergence guarantees no longer hold. However, in many real-world problems, where the control is decentralized, there is no way to avoid a multiagent RL formulation, as for instance the scenario we deal with in the present paper, namely control of traffic signals, whose basic concepts we briefly review next.

2.2. RL-Based Traffic Signal Control

Besides safety and other issues, one aim of a traffic signal controller is to decide on a split of green times among the various phases that were designed to deal with geometry and flow issues at an intersection. This can be done in several ways (for more details, please see a textbook such as [5]). In this paper, the controller is given a set of phases and has to decide which one will receive right of way (green light).

A phase is defined as a group of non-conflicting movements (e.g., flow in two opposite traffic directions) that can have a green light at the same time without conflict.

In its simplest form, the control is based on fixed times, whose split of the green time among the various phases can be computed based on historical data on traffic flow, if available. The problem with this approach is that it cannot adapt to changes in the traffic demand. This may lead to an increase in the waiting time. To mitigate this problem, it is possible to use an adaptive scheme and thus give priority to lanes with longer queues (or other measures of performance).

Adaptive approaches based on RL were developed, as we discuss at the beginning of the next section.

3. Related Work

In the last two decades there has been a large body of work that proposes the use of RL for traffic signal control. Given the extension and diversity of such a body of work, it is outside the scope of the present paper to review them. Readers are directed to surveys such as [6, 7, 8, 9]. In any case, those surveys show that there has already been a significant contribution of RL techniques to control traffic signals. However, issues of scalability and performance remain open, especially if tabular methods are used, such as the aforementioned Q-learning. Tabular methods require discretization of the state space. The finer such discretization is, the poorer the computational performance of the learning task, since agents need to visit an increasing number of states.

Therefore, the remaining of this section discusses alternative ways to tackle these issues.

A first line of research does use tabular methods, with various levels of discretization of the state space, thus depicting different levels of performance of the learning task. In this class, well-known works include [10, 11, 12, 13], among others.

A second class of works avoids using tabular methods such as Q-learning. Rather, they propose the use of function approximation. For instance, [14] uses tile coding. Recently, many studies have used deep neural networks to approximate the Q-function (e.g., DQN [15]). However, non-linear function approximation is known to diverge in multiple cases [16, 17]. In order to address these shortcomings, [18] proposed the use of linear function approximation, which has guaranteed convergence and error bounds.

A third research line employs clustering methods together with some sort of RL aproach in order to tackle the large or continuous state space. However, to the best of our knowledge, there has been no attempt to employ clustering-based methods to traffic signal control. Applications span over obstacle avoidance [19]; games such as Atari [20]; and for partitioning the state space in general [21]. Also worth mentioning are works that employ hierarchical clustering and/or hierarchical RL for state abstraction or experience generalization, such as [22, 23, 24, 25, 26, 27, 28, 29, 30]. Again, these works cover applications other than traffic signal control and deal with single agent scenarios.

Finally, another way to tackle the issues that arise from large and continuous state spaces appeared in [2, 3], in which a temporal difference learning algorithm based on k-nearest neighbors was presented. RL approaches based on this technique have been used in different domains, like robot motion control [31] and video streaming [32]. However, to our best knowledge, its application in traffic signal control, as we do here, is novel. Moreover, as we tackle a network of signal controllers, our work deals with multiagent reinforcement learning. Even if the agents learn independently, because a multiagent setting is ineherently non-stationary, this makes the learning task much more challenging. More details about these challenges are discussed in [6].

In a nutshell, our paper intends to help fill a gap in the traffic signal control by means of RL. We accomplish this by applying a method that avoids the need of discretizing continuous states, and also without requiring a function approximation technique (which makes the learning

potentially more difficult to manage and understand [17]).

4. Materials and Methods

In many real world problems, the state space that is associated with an RL task is large and continuous. For example, this is the case when the task is to control a traffic signal, where the state refers to queue length, density of vehicles, or a combination of both (please refer to [8] for a discussion about popular formulations of the state space for this particular domain). Therefore, the quality of the learning task depends on how the state is discretized. Also, learning agents must be able to effectively adapt to a changing environment.

This section explains the approach we employ in order to address the just mentioned issues related to how to deal with a continuous state space. Also, we discuss how the method was applied to controlling a network of traffic signals.

4.1. Temporal Difference Learning Based on k-Nearest Neighbors

Essentially, the method estimates the Q-values of the current state by calculating the weighted average of the Q-value estimates of the k nearest states, based on the euclidean distance metric. The closer a neighbor state is, the greater the impact its Q-value estimates have on the Q-values of the current state. It then selects an action based on an exploration strategy like ε-greedy, transitions to a new state and receives a reward. Subsequently, it calculates a temporal difference (TD) error based on the reward and the expected value of the last and new state, and uses this error to update the Q-value estimates of all the k nearest states that contributed to estimate the Q-values of the previous state.

4.1.1. Estimating Q-Values

Each dimension of the state space could be of a different order of magnitude. When measuring the Euclidean distance between two states, dimensions of a greater order of magnitude could introduce a selection bias, and impact more heavily the value of the distance than dimensions of a smaller order of magnitude. To prevent this bias, when a state s_t is observed at time step t, it is normalized, according to Eq. (2), where s_m and s_M denote the lower and upper bounds of the state space, respectively.

$$\hat{s_t} = 2 \cdot (\frac{s_t - s_m}{s_M - s_m}) - 1 \tag{2}$$

The agent keeps a record of each visited state and an estimate of the Q-values for each of these states. After normalizing the observed state s_t, the k nearest previously visited states according to the euclidean distance metric are selected, in order to determine the k-nearest neighbors set. A weight is then calculated for each state in the set, according to Eq. (3), where w_i and d_i represent the weight of the i_{th} nearest state and the euclidean distance between s_t and the i_{th} nearest state, respectively, and $\mathcal{K}$ represents the k-nearest neighbors set.

$$w_i = \frac{1}{1 + d_i^2} \ , \ \forall i \in \mathcal{K} \tag{3}$$

The Q-value of a state-action pair is determined by the estimate of an expected value, in which the probabilities of each state in the $\mathcal{K}$ set are given by Eq. (4), where $p(i)$ is the probability of the i_{th} nearest state in the $\mathcal{K}$ set.

$$p(i) = \frac{w_i}{\sum_{j \in \mathcal{K}} w_j} \ , \ \forall i \in \mathcal{K} \tag{4}$$

For each action a, the Q-value of the state-action pair (s_t, a) is then determined according to an estimate of expected value using the probabilities and the current estimates of the Q-values of each state in the k-nearest neighbors set, according to Eq. (5).

$$Q(s_t, a) = \sum_{i \in \mathcal{K}} p(i) Q(i, a) \tag{5}$$

This way, the method avoids both state discretization and function approximation, as it estimates the Q-values of the current state by calculating the weighted average of the Q-values of the nearest previously visited states.

4.1.2. Action Selection and Updating Q-Values

Having the expected value of each action for the current state s_t, an exploration strategy, such as ε-greedy, is used to select an action a_t. After taking the selected action, the agent transitions to a new state s_{t+1} and receives a reward r_t. In order for learning to occur, the agent estimates the expected value of taking each possible action in the next state, s_{t+1}, via the k-NN approach explained in Section 4.1.1 (using Eq. (2), Eq. (3), Eq. (4) and Eq. (5) on s_{t+1}), and calculates the TD error δ (which is the basic update rule of a TD learning method, obtained by measuring the difference between the estimated value of a state or a state-action pair, and the improved estimate obtained after gaining more experience), using Eq. (6).

$$\delta = r_t + \gamma \max_a (Q(s_{t+1}, a)) - Q(s_t, a_t) \tag{6}$$

The Q-value estimate for taking the action a_t in each state in the k-nearest neighbors set is then updated using Eq. (7).

$$Q(i, a_t) \leftarrow Q(i, a_t) + \alpha \delta p(i) \ , \ \forall i \in \mathcal{K} \tag{7}$$

A pseudocode for the algorithm is presented in Algorithm 1.

4.2. Scenario and Formulation of the RL task

As aforementioned, our goal is to present a multiagent RL approach based on k-nearest neighbors as an alternative to deal with large and continuous state spaces in traffic signal control. To measure the performance of our approach, we use a scenario that contains a network of traffic signal controllers (or agents), introduced in [33]. Each of these agents independently acts and learns using the k-NN method.

The scenario selected is especially interesting and non-trivial because, besides containing a network of intersections, it considers changes in the environment, i.e., changes in traffic

Algorithm 1: Temporal Difference Learning Based on k-Nearest Neighbors

input : α, γ, ε, k
foreach *episode* **do**
  Initialize s;
  foreach *step of episode* **do**
    Normalize s using Eq. (2);
    $\mathcal{K} \leftarrow k$ nearest neighbors of $\hat{s}$;
    Calculate state weights using Eq. (3);
    Calculate state probabilities using Eq. (4);
    Estimate Q-values of s using Eq. (5);
    Choose action a from s using Q-values;
    Take action a, receive r and observe s';
    Calculate TD error using Eq. (6);
    Update Q-values of states in $\mathcal{K}$ using Eq. (7);
    $s \leftarrow s'$;
  end
end

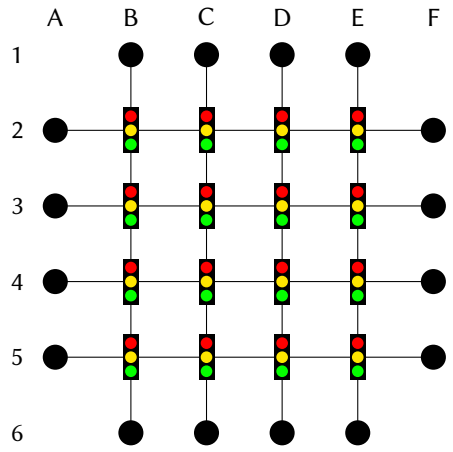

Figure 1: 4×4 grid network.

contexts. We stress that such changes are rarely addressed in RL-based methods for signal control. Further, we compare our results to those reported in [33], in which a tabular method (Q-learning) with a fixed discretization scheme was used, which included different features.

In short, while [33] has reported results that outperformed other baselines in a non-trivial environment (traffic situation changes), our aim here is to show that a better way of treating the continuous state space may pay off.

Our experiments were performed using a microscopic traffic simulator, namely SUMO [34] (Simulation of Urban MObility). The scenario is a 4x4 traffic signal grid, with 16 traffic signal controllers (one in each intersection), which are the learning agents. In this scenario, agents are

homogeneous, i.e., they have the same set of available actions.

Figure 1 shows the traffic grid network's topology. We defer the details about the trip demands; these are discussed in Section 4.3, where we show how the various flows of vehicles change along the simulation time.

Every link has 150 m in length, two lanes and is one-way. There are four vertical (B1 $\rightarrow$ B6, C1 $\rightarrow$ C6, D1 $\rightarrow$ D6, E1 $\rightarrow$ E6) and four horizontal (A2 $\rightarrow$ F2, A3 $\rightarrow$ F3, A4 $\rightarrow$ F4, A5 $\rightarrow$ F5) Origin-Destination (OD) pairs. A vehicle is inserted in an origin node, and is removed from the simulation in a destination node. Vehicles go in the North-South (N-S) direction in vertical links, and in the West-East (W-E) direction in horizontal links.

Traffic signals in our scenario have a minimum and maximum time they must remain green. They are referred to as *minGreenTime* and *maxGreenTime*, respectively.

We compare our RL method to a baseline (from [33]). As is often with RL problems, we use a multiagent MDP (MMDP) to formalize our traffic signal control problem. Thus, besides the set of agents, we need to define the other sets and functions that compose the MDP. In the next sections we define the state space, the action space, and the reward function.

4.2.1. State Space

At each time step t (which corresponds to five seconds of real-life traffic dynamics), each agent observes a vector s_t, which describes the current state of the respective intersection. We use the default state definition in [35], shown in Eq. (8), where $\rho_1 \in \{0, 1\}$ and $\rho_2 \in \{0, 1\}$ are binary variables that indicate the current active green phase (see Section 2.2 for a description about phases). $g \in \{0, 1\}$ is a binary variable that indicates whether or not the current green phase has been active for more than *minGreenTime*. L is the set of all incoming lanes. The density $\Delta_l \in [0, 1]$ is defined as the number of vehicles in the incoming lane $l \in L$ divided by the total capacity of the lane. $q_l \in [0, 1]$ is defined as the number of queued vehicles in the incoming lane $l \in L$ divided by the total capacity of the lane. A vehicle is considered to be queued if its speed is below 0.1 m/s.

$$s_t = [\rho_1, \rho_2, g, \Delta_1, ..., \Delta_{|L|}, q_1, ..., q_{[L]}] \tag{8}$$

Note that although it is common in the literature that only one feature be used (i.e., either density or queue), here we employ both as this was the case in [33], which we use as a baseline for comparison.

4.2.2. Action Space

Each learning agent chooses a discrete action a_t at each time step t. For our scenario, since all intersections have two incoming links, there are two phases, so each agent has only two actions: *keep* and *change*. The former keeps the current green signal active, while the latter switches the current green light to another phase. The agents can only choose *keep* if the current green phase has been active for less than *maxGreenTime*, and can only choose *change* if the current green phase has been active for more than *minGreenTime*.

4.2.3. Reward Function

As with the state space, we also use the default reward function given in [33], which is the cumulative vehicle delay, shown in Eq. (9), where D_t is the cumulative vehicle delay at the intersection at time step t.

$$r_t = D_t - D_{t+1} \tag{9}$$

We define the cumulative vehicle delay at time step t, D_t, as being the sum of the waiting time of all of the vehicles (a vehicle is waiting if its speed is less than 0.1 m/s) that are approaching the intersection. This is calculated as in Eq. (10), where V_t is the set of incoming vehicles, and d_t^v is the delay of vehicle v at time t.

$$D_t = \sum_{v \in V_t} d_t^v \tag{10}$$

4.3. Changing the Demand Along the Simulation Horizon

As mentioned, a particular challenge in the present work is that the demand changes during the simulation time, i.e., the amount of vehicles traveling from a given origin to a given destination changes from time to time. Following [33], we also denote these different situations by *traffic contexts* or simply *context*, and refer to the grid depicted in Figure 1. There are two different traffic contexts, which correspond to two different vehicle flow rates:

- Context 1 (NS = WE): one vehicle is inserted in all 8 Origin-Destination pairs every 3 seconds.
- Context 2 (NS $<$ WE): one vehicle is inserted every 6 seconds in the 4 OD pairs in the North-South direction and one vehicle is inserted every 2 seconds in the 4 OD pairs in the West-East direction.

Each simulation runs for 80,000 seconds and switches contexts every 20,000 seconds; thus there are three changes in context (Context 1 $\rightarrow$ Context 2 $\rightarrow$ Context 1 $\rightarrow$ Context 2).

4.4. Signal Control and Learning Parameters

The value used for *minGreenTime* was 10 seconds, and 50 seconds for *maxGreenTime*. Also, as aforementioned, one simulation time step corresponds to five seconds of real-life traffic dynamics, as in [33].

We also kept the same parameter values that were used in that paper. Thus: α = 0.1, γ = 0.99 and ϵ = 0.05. Regarding the k-NN method, k = 200 was used. Several values for k were tested, and 200 was selected because this brought the best results.

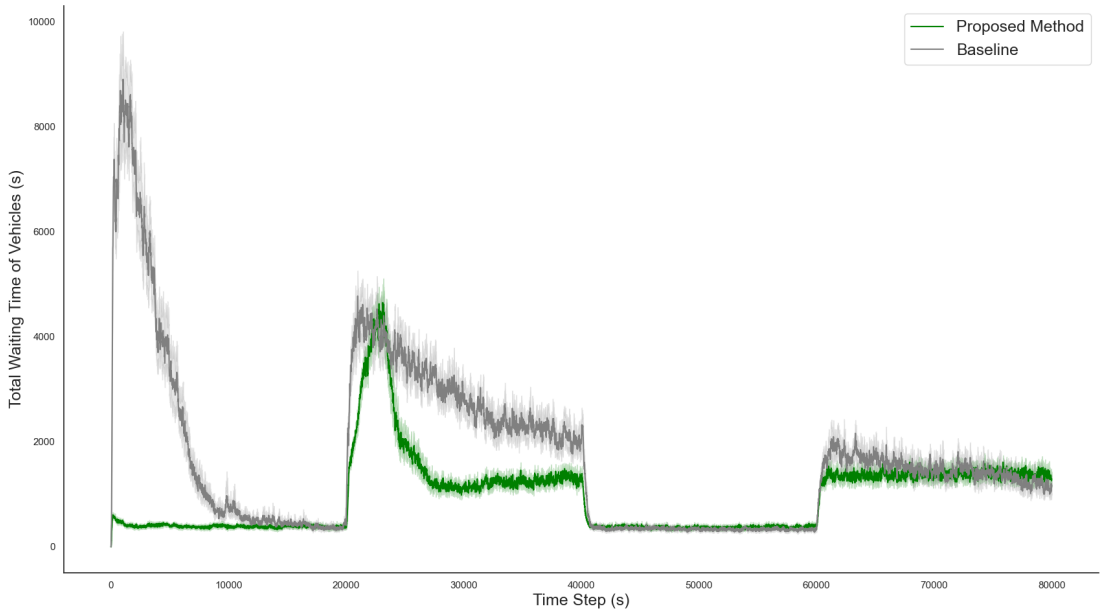

Figure 2: Comparison between the baseline and the proposed method. Waiting time over all vehicles (average and deviation over 20 repetitions), along time.

5. Results and Discussion

Next we discuss the simulation results of our proposed method. To evaluate our approach, we calculate the average waiting time of all vehicles, which is a metric commonly used in the literature. As mentioned previously, a vehicle whose speed is less than 0.1 m/s is considered to be waiting. Reducing the average waiting time of vehicles is an important objective for traffic signal control, as it reduces vehicle queue sizes and improves traffic flow.

Figure 2 shows the average waiting time (over all vehicles) for our method and the baseline, over 20 repetitions, along the 80,000 seconds of simulated real-world traffic dynamics, which represent 16,000 actions taken (since the agent takes an action every 5 seconds). The vehicle flow also changed at 20,000 seconds, 40,000 seconds and 60,000 seconds (as aforementioned, the traffic contexts changed in the following manner: Context 1 $\rightarrow$ Context 2 $\rightarrow$ Context 1 $\rightarrow$ Context 2).

Comparing the proposed method with the baseline in [33], it is clear that our approach outperforms the baseline, converging sooner. The first time the agents explore each of the two contexts (from 0 to 20,000 seconds for Context 1, and from 20,000 seconds to 40,000 seconds for Context 2), in both of them, the proposed method converged much faster than the baseline.

In Context 1, the proposed method quickly finds an approximate best solution requiring very little exploration, while the baseline takes a considerably greater amount of steps before converging. This is the simplest context, since a vehicle is inserted every 3 seconds in all OD pairs, so there is a symmetry to the traffic flow. The results for this context indicate that for simpler scenarios, the approach based on k-NN is vastly superior, since it is able to rapidly

detect patterns within the agent's dataset of experiences.

Context 2 involves a more difficult vehicular flow, since vehicles are inserted unevenly (one vehicle is inserted every 2 seconds in the 4 West-East OD pairs, and one vehicle is inserted every 6 seconds in the 4 North-South OD pairs). Because more vehicles flow in the West-East direction, larger queues are formed in that direction and the traffic controllers must adapt to this new situation. As can be seen, both the baseline and the proposed method have a decline in performance, as they are dealing with a context they never saw before, but our proposed method quickly reduces the average waiting time, taking around 8,000 seconds (1,600 time steps) in this new context to do so, while the baseline only reaches this level the second time the context changes to Context 2.

For the second half of the simulation (from 40,000 seconds to 80,000 seconds), the context changes back to Context 1, and then changes back to Context 2. As can be seen, both methods at this point have already approximately converged, and are operating at a similar level of performance in terms of waiting time. However, an advantage of our method is that it shows significantly less standard deviation in comparison to Q-learning.

6. Conclusion and Future Work

In this paper, a multiagent system with agents learning via a temporal difference learning algorithm based on k-nearest neighbors was presented as an option to control traffic signals in real-time. This approach seeks to solve the issues that arise when the state space is continuous and thus, a poor discretization leads to poor performance. Or, rather, when the discretization is fine, but then leads to scalability issues. The goal here was to employ a method that could deal with a continuous state space and still be efficient.

By simultaneously avoiding the need for state discretization and the usage of function approximation, the approach is able to somewhat circumvent significant issues that arise in the application of RL for traffic signal control. This was shown by simulating the method in a complex scenario that involved not only many traffic signals, but also changes in the flow of vehicles (traffic contexts).

In this scenario, a tabular method that includes more than one feature was used as a baseline for comparison. Our results showed that our method outperformed the baseline; specifically, it converges earlier to a situation in which the waiting time of vehicles is lower.

To our best knowledge, this is the first time that a RL approach combined with k-nearest neighbors was used for traffic signal control. Also, the scenario used involved changes in traffic contexts, which has been rarely discussed in the literature.

As a next step, the proposed method will be combined with an approach that incrementally clusters the agent's experiences and uses cluster fusion to share knowledge among agents in order to improve even more the performance of the traffic signal controllers. As for extension of the experiments, we intend to use scenarios in which agents are heterogeneous with regard to having a different set of actions or phases.

Acknowledgments

This work is partially supported by FAPESP and MCTI/CGI (grants number 2020/05165-1 and 2021/05093-3), and by a FAPERGS grant (Vicente N. de Almeida). Ana Bazzan is partially supported by CNPq under grant number 304932/2021-3, and by the German Federal Ministry of Education and Research (BMBF), Käte Hamburger Kolleg Cultures des Forschens/ Cultures of Research.

References

[1] M. Papageorgiou, Traffic control, in: R. W. Hall (Ed.), Handbook of Transportation Science, Kluwer Academic Pub, 2003, pp. 243–277.

[2] J. A. Martín H., J. de Lope, A k-NN based perception scheme for reinforcement learning, in: R. M. Díaz, F. Pichler, A. Q. Arencibia (Eds.), Computer Aided Systems Theory - EUROCAST 2007, volume 4739 of *Lecture Notes in Computer Science*, Springer, Berlin, Heidelberg, 2007, pp. 138–145. doi:10.1007/978-3-540-75867-9_18.

[3] J. A. Martín H., J. de Lope, D. Maravall, The kNN-TD reinforcement learning algorithm, in: J. Mira, J. M. Ferrández, J. R. Álvarez, F. de la Paz, F. J. Toledo (Eds.), Methods and Models in Artificial and Natural Computation. A Homage to Professor Mira's Scientific Legacy, Springer, Berlin, Heidelberg, 2009, pp. 305–314. doi:10.1007/978-3-642-02264-7_32.

[4] C. Watkins, Learning from Delayed Rewards, Ph.D. thesis, University of Cambridge, 1989.

[5] R. P. Roess, E. S. Prassas, W. R. McShane, Traffic Engineering, 3rd ed., Prentice Hall, 2004.

[6] A. L. C. Bazzan, Opportunities for multiagent systems and multiagent reinforcement learning in traffic control, Autonomous Agents and Multiagent Systems 18 (2009) 342–375. doi:10.1007/s10458-008-9062-9.

[7] K.-L. A. Yau, J. Qadir, H. L. Khoo, M. H. Ling, P. Komisarczuk, A survey on reinforcement learning models and algorithms for traffic signal control, ACM Comput. Surv. 50 (2017). doi:10.1145/3068287.

[8] H. Wei, G. Zheng, V. V. Gayah, Z. Li, A survey on traffic signal control methods, 2020. URL: http://arxiv.org/abs/1904.08117, preprint arXiv:1904.08117.

[9] M. Noaeen, A. Naik, L. Goodman, J. Crebo, T. Abrar, B. Far, Z. S. H. Abad, A. L. C. Bazzan, Reinforcement learning in urban network traffic signal control: A systematic literature review, 2021. URL: engrxiv.org/ewxrj. doi:10.31224/osf.io/ewxrj.

[10] M. Aslani, M. S. Mesgari, M. Wiering, Adaptive traffic signal control with actor-critic methods in a real-world traffic network with different traffic disruption events, Transportation Research Part C: Emerging Technologies 85 (2017) 732–752. doi:https://doi.org/10.1016/j.trc.2017.09.020.

[11] P. Balaji, X. German, D. Srinivasan, Urban traffic signal control using reinforcement learning agents, IET Intelligent Transportation Systems 4 (2010) 177–188.

[12] S. El-Tantawy, B. Abdulhai, H. Abdelgawad, Multiagent reinforcement learning for integrated network of adaptive traffic signal controllers (marlin-atsc): Methodology and large-scale application on downtown toronto, Intelligent Transportation Systems, IEEE Transactions on 14 (2013) 1140–1150. doi:10.1109/TITS.2013.2255286.

[13] H. Wei, G. Zheng, H. Yao, Z. Li, Intellilight: A reinforcement learning approach for intelligent traffic light control, in: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '18, Association for Computing Machinery, New York, NY, USA, 2018, p. 2496–2505. doi:10.1145/3219819.3220096.

[14] M. Abdoos, N. Mozayani, A. L. Bazzan, Hierarchical control of traffic signals using Q-learning with tile coding, Appl. Intell. 40 (2014) 201–213. doi:10.1007/s10489-013-0455-3.

[15] E. Van Der Pol, Deep Reinforcement Learning for Coordination in Traffic Light Control, Ph.D. thesis, University of Amsterdam, 2016.

[16] L. Baird, Residual algorithms: Reinforcement learning with function approximation, in: In Proceedings of the Twelfth International Conference on Machine Learning, Morgan Kaufmann, 1995, pp. 30–37.

[17] R. S. Sutton, A. G. Barto, Reinforcement learning: An introduction, second ed., The MIT Press, 2018.

[18] L. N. Alegre, T. Ziemke, A. L. C. Bazzan, Using reinforcement learning to control traffic signals in a real-world scenario: an approach based on linear function approximation, IEEE Transactions on Intelligent Transportation Systems (2021). doi:10.1109/TITS.2021.3091014.

[19] A. Matt, G. Regensburger, An adaptive clustering method for model-free reinforcement learning, in: Proceedings of INMIC 2004, IEEE, 2004, pp. 362–367.

[20] X. Ma, S.-Y. Zhao, W.-J. Li, Clustered reinforcement learning, 2019. ArXiv preprint arXiv:1906.02457.

[21] S. Mannor, I. Menache, A. Hoze, U. Klein, Dynamic abstraction in reinforcement learning via clustering, in: Proceedings of the twenty-first international conference on Machine learning (ICML), ACM Press, New York, NY, USA, 2004, pp. 71–79.

[22] R. Akrour, F. Veiga, J. Peters, G. Neumann, Regularizing reinforcement learning with state abstraction, in: 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE, 2018, pp. 534–539.

[23] A. Dereventsov, R. Vatsavai, C. G. Webster, On the unreasonable efficiency of state space clustering in personalization tasks, in: 2021 International Conference on Data Mining Workshops (ICDMW), IEEE, 2021, pp. 742–749.

[24] N. Entezari, M. E. Shiri, P. Moradi, A local graph clustering algorithm for discovering subgoals in reinforcement learning, in: Communication and Networking, Springer Berlin Heidelberg, 2010, pp. 41–50. doi:10.1007/978-3-642-17604-3_5.

[25] X. Guo, Y. Zhai, K-means clustering based reinforcement learning algorithm for automatic control in robots, International Journal of Simulation Systems, Science and Technology 17 (2016). doi:10.5013/IJSSST.a.17.24.06.

[26] M. Hashemzadeh, R. Hosseini, M. N. Ahmadabadi, Clustering subspace generalization to obtain faster reinforcement learning, Evolving Systems 11 (2020) 89–103.

[27] G. Kheradmandian, M. Rahmati, Automatic abstraction in reinforcement learning using data mining techniques, Robotics and Autonomous Systems 57 (2009) 1119–1128. doi:10.1016/j.robot.2009.07.002.

[28] L. Lehnert, M. L. Littman, Transfer with model features in reinforcement learning, 2018. ArXiv preprint arXiv:1807.01736.

[29] A. I. Panov, A. Skrynnik, Automatic formation of the structure of abstract machines in hierarchical reinforcement learning with state clustering, 2018. ArXiv preprint arXiv:1806.05292.

[30] A. Srinivas, R. Krishnamurthy, P. Kumar, B. Ravindran, Option discovery in hierarchical reinforcement learning using spatio-temporal clustering, 2016. ArXiv preprint arXiv:1605.05359.

[31] F. Han, L. Jin, Y. Yang, Z. Cao, T. Zhang, Research on robot motion control based on local weighted kNN-TD reinforcement learning, in: Proceedings of the 10th World Congress on Intelligent Control and Automation, 2012, pp. 3648–3651. doi:10.1109/WCICA.2012.6359080.

[32] H. Lin, Z. Shen, H. Zhou, X. Liu, L. Zhang, G. Xiao, Z. Cheng, KNN-Q learning algorithm of bitrate adaptation for video streaming over HTTP, in: 2020 Information Communication Technologies Conference (ICTC), 2020, pp. 302–306. doi:10.1109/ICTC49638.2020.9123312.

[33] L. N. Alegre, A. L. C. Bazzan, B. C. da Silva, Quantifying the impact of non-stationarity in reinforcement learning-based traffic signal control, PeerJ Computer Science 7 (2021) e575. URL: http://dx.doi.org/10.7717/peerj-cs.575. doi:10.7717/peerj-cs.575.

[34] P. A. Lopez, M. Behrisch, L. Bieker-Walz, J. Erdmann, Y.-P. Flötteröd, R. Hilbrich, L. Lücken, J. Rummel, P. Wagner, E. Wießner, Microscopic traffic simulation using SUMO, in: The 21st IEEE International Conference on Intelligent Transportation Systems, 2018.

[35] L. N. Alegre, SUMO-RL, https://github.com/LucasAlegre/sumo-rl, 2019.