Traffic light control using multiagent reinforcement learning

Miguel Ângelo Mendes Coelho

Instituto Superior Técnico
University of Lisbon,
Av. Rovisco Pais, 1 1049-001 Lisboa, Portugal,
miguelmendescoelho@gmail.com

Abstract. The existing traffic infrastructure cannot handle the current traffic demand in large city traffic networks. Previous works based on multi-agent reinforcement learning that exploit space locality using coordination graphs have shown promising results. However, the effectiveness of exploiting time locality using sparse interactions is unknown. This work presents a comparative study between multi-agent methods that use coordination graphs and sparse interactions in realistic traffic scenarios.

Keywords: traffic light control · reinforcement learning · multi-agent systems

Table of Contents

1 Introduction

The latest growth in population led to an increase in social and economic activities that then resulted in a higher demand for transportation. The effectiveness of the current transportation systems is crucial to ensure short travel times of individuals and goods while also having minimal impact on the ecological environment. This high transportation demand has outclassed the existing traffic infrastructures that are now incapable of handling everyday traffic congestions. The present traffic infrastructure uses traffic lights, signaling and transit planning to minimize these congestions, where traffic lights are the most typical control mechanism.

One way of increasing the traffic capacity is to expand the current road infrastructure. This is normally avoided due to its monetary cost, extensive time commitment and environmental constraints.

Another approach is to create Intelligent Transportation Systems (ITS). ITS take advantage of the data collected by sensors to provide traffic management, specifically traffic lights, that can adapt to different traffic volumes and unexpected disruptions to raise the current traffic capacity. In practice, most traffic lights simply alternate green and red lights in fixed intervals. This type of controllers are referred as fixed timing systems. The duration may change depending on the time of day, for example, in peak hours, but otherwise these are not optimized, specially for city level traffic flow.

Some of the current traffic lights actually use the information from external sensors by changing their phase depending on data collected from the sensors. These are more flexible than the fixed timing controllers, however, they only handle data from the intersection they work on and can even disturb the traffic light cycles of other neighboring intersections.

Machine learning algorithms constitute an effective way to build ITS since they can use sensory data to create models that adapt to the environment and improve the traffic flow without human intervention. Reinforcement learning (RL) is a machine learning paradigm where an agent learns the best actions to take in different states by experimenting with the environment. RL is well suited to solve the traffic light control problem since we can model the traffic controller as the agent, the traffic data as the environment and the phase timings as the actions.

Some works [1], [2] have applied reinforcement learning to solve traffic control problems, using the growing available traffic data to outperform classical approaches like fixed and actuated timing controllers. However, most of the settings are oversimplified, e.g, single intersection scenarios, and cannot be easily compared between each other.

When considering more complex environments with several inter-connected intersections, single agent and uncoordinated multi-agent models fail to obtain good traffic controllers due to the exponential increase in complexity of the state space and lack of coordination between agents.

In previous works that use coordinated multi-agent models, the space locality of interaction between agents is exploited by the use of coordination graphs that limit how the agents interact, e.g, agents only interact with other nearby agents. However, we can restrict the interactions further by assuming that the agents only need to coordinate at certain times, creating a model with sparse interactions.

Thus rises the main question of this work: Does the exploitation of sparse interactions in multi-agent RL for traffic control offer performance advantages over the more classical approaches based on coordination graphs?

1.1 Objectives

The objective of the current work is to develop multi-agent reinforcement learning traffic controllers that exploit space and time locality assumptions to understand their traffic performance and computation time.

Specifically, this work aims to:

- Prepare the simulator that will be used to train and evaluate the RL controllers and create a set of realistic training scenarios. In particular, a 3x1 arterial network and two grid networks with increasing size (6 and 21 traffic controllers), further explained in Section 4.1.
- Develop the baseline traffic controllers that will be used to compare with the other RL controllers, in particular, a fixed time controller using Websters method and a uncoordinated tabular Q-learning controller.
- Develop RL traffic controllers that exploit locality, in particular, a coordination graph approach based in the Max-plus algorithm that exploits space locality and traffic controller that uses sparse interactions, exploiting time locality, both further explained in Section 2.3.
- Compare the RL controllers that exploit these localities using a set of evaluations metrics, further detailed in Section 4.3, to access the performance of each controller.

1.2 Outline

The remainder of this document is organized as follows: Section 2 covers some traffic light control terms and objectives as well as single and multi-agent reinforcement learning basics. Section 3 explores the different approaches that have been used to solve this problem, namely, fixed timing and RL based controllers. Section 4 details how the previously mentioned objectives will be accomplished and, finally, Section 6 recaps the problem and solution presented in this work.

2 Background

This Section covers some traffic control concepts that will be used in the latter Sections, followed by the background on single-agent reinforcement learning that is modeled as a Markov Decision Process. Section 2.3 presents an extension of this model for multi-agent systems which will be relevant for the multi-agent RL traffic controllers presented in this work.

2.1 Traffic light control

In this Section, some traffic light control terms and the traffic control objective are defined.

2.1.1 Traffic term definition

A traffic network can be represented by a graph where junctions are the nodes and roads are the edges, where each road contains a certain amount of lanes. An intersection is a type of junction where two roads intersect and is where traffic light controller are usually placed. The edges of a road that meet the intersection are labeled approaches and can be defined as incoming or outgoing depending if the cars in that lane are entering or leaving the intersection, respectively.

Figure 1 represents a traffic intersection with two incoming and outgoing north-south approaches, each with two lanes, and two incoming and outgoing east-west approaches, each with three lanes.

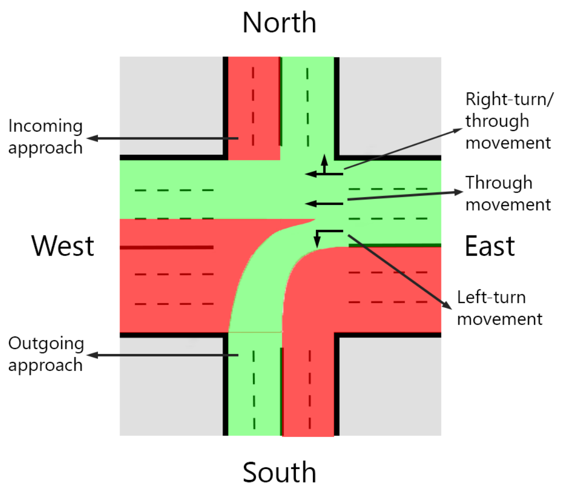


Fig. 1. Traffic intersection with a phase that prevents vehicle collision.

A traffic movement consists of vehicles moving from lane in an incoming approach to another lane in an outgoing approach. These are usually defined as right-turn, through, left-turn or combinations of the previous three.

A movement signal is defined as a traffic movement that is allowed by the traffic controller. Figure 1 shows three valid movement signals, right-turn/through, through and left-turn. For example, in the left-turn movement lane cars can only go to the outgoing south approach.

A phase is a combination of valid movement signals. Two movement signals are valid if they can both be set to green without allowing vehicle collision. The three movements represented in figure 1 are a valid phase.

A signal plan is a sequence of phases and their respective duration, or equivalently, start time. The time that takes to cycle through every phase in a signal plan is defined as cycle length.

2.1.2 Traffic control objectives

The ultimate objective in a traffic network is to make vehicle movement more efficient and safe. Safety is mostly insured though phases with valid movements. Efficiency is usually quantified by different metrics, such as:

– Travel time: the average travel time of all vehicles in the network. In other words, the average time since each vehicle entered and exited the network.
– Cumulative Delay: the cumulative sum of the time a vehicle as stopped in any intersection.
– Number of stops: the number of stops for all vehicles.
– Throughput: the number of vehicles that traverse one or more intersections in a period of time, for example, in the last phase.
– Queue Length: the sum of the number of waiting vehicles in a set of lanes or a phase.

Thus, the objective is to minimize one or more of the previous metrics. The most important and most used metric is travel time since it is strongly related to congestion, has the same meaning in all transportation modes and even those not familiar with traffic control terms can comprehend the concept. To evaluate the RL controllers developed in this work, the travel time, cumulative delay and number of stops will be used. A more detailed explanation of this evaluation is done in Section 4.3.

2.2 Single-Agent RL

The reinforcement learning problem, introduced by Sutton et al. [3] can be modeled as a Markov Decision Process (MDP). An MDP is a sequential decision making model, where an agent iteratively observes the environment, decides on an action to take and analyzes the reward for the selected action.

We can formally define a MDP with the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$:

– $\mathcal{S} = \{s_1, ..., s_n\}$ is the finite set of environmental states.
– $\mathcal{A} = \{A(s_t),\ s_t \in \mathcal{S}\}$ where $A(s_t)$ is the finite set of all possible actions in the state s_t.
– $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$ is a stochastic transition probability function, where $\mathcal{P}(s_t, a_t, s_{t+1})$ defines the probability of moving to state s_{t+1} after choosing action a_t in state s_t.

– $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, where $\mathcal{R}(s_t, a_t)$ defines the expected reward the agent will receive after choosing action a_t in state s_t.
– $\gamma \in [0, 1]$ is the discount factor, that represents how much future rewards are valued when compared to present rewards.

At each time step, t, the agent observes the current state of the environment, $s_t \in \mathcal{S}$ and chooses an action $a_t \in A(s_t)$. This action causes a change in the environment state, defined by the transition probability function $\mathcal{P}(s_t, a_t, s_{t+1})$ and the agent receives a numerical reward defined by $\mathcal{R}(s_t, a_t)$ that represents the quality of the action taken. This ecosystem is illustrated in Figure 2.

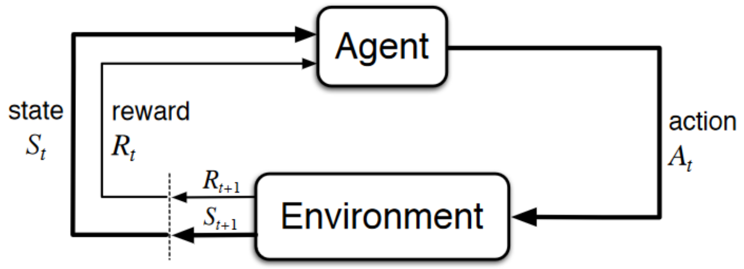


Fig. 2. The interaction loop between the environment and the agent [3].

In the reinforcement learning problem, the transition probability function, $\mathcal{P}$, and the reward function, $\mathcal{R}$, are unknown so the agent needs to find the optimal policy by observing (s_t, a_t, s_{t+1}, r_t) tuples that are collected by trial and error interaction with the environment.

A policy is the mapping between actions and states, $\pi : \mathcal{A} \times \mathcal{S} \to [0, 1]$ where $\pi(a_t|s_t)$ is the probability of selecting action a_t when the agent is in state s_t.

The objective of the agent is to converge to an optimal policy π^* that maps the states to the best actions to take, maximizing the expected discounted reward over time.

Formally, the optimal policy is defined as:

$$\pi^* = \arg\max_{\pi} \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t \mathcal{R}(s_t, a_t) \middle| s_0 = s \right], \forall s \in \mathcal{S} \tag{1}$$

To converge to the optimal policy, RL algorithms rely on the value function, $V^\pi : \mathcal{S} \to \mathbb{R}$ and the Q-value function, $Q^\pi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$.

The value function represents the expected reward of starting in state s and following the policy π afterwards and is defined as:

$$V^\pi(s) = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \middle| s_t = s \right] \tag{2}$$

The Q-value function is similar to (2) but defined for states and actions. This function estimates the expected reward of choosing action a in state s and following the policy π afterwards. It is defined as:

$$Q^{\pi}(s,a) = \mathbb{E}_{\pi}\left[\sum_{k=0}^{\infty}\gamma^{k} r_{t+k+1}\middle| s_t = s, a_t = a\right] \tag{3}$$

For example, the tabular Q-learning algorithm [4] estimates the Q-values directly using:

$$Q(s_t,a_t) = Q(s_t,a_t) + \alpha\left[r_{t+1} + \gamma \max_{a} Q(s_{t+1},a) - Q(s_t,a_t)\right] \tag{4}$$

All of the previous definitions regarding MDPs were defined for a single learning agent. This approach can be extended to multiple learning agents in different ways. A simple approach would be to make agents ignore other agents, thus treating the problem as multiple single-agent MDPs. This is usually not adequate since the actions of one agent influence the state or reward of other agents. Coordination is needed so agents can work together to maximize a common goal.

2.3 Multi-Agent RL

The MDP framework can be extended to multi-agent systems by using Stochastic Games (SG). In stochastic games, the actions of the agents now form a joint decision based on environment information of all the agents. A stochastic game can be defined by the tuple $(\mathcal{S},\mathrm{U},\mathcal{P},\mathcal{R}_{1...n},\gamma)$, where:

- n is the number of agents in the system.
- $\mathcal{S} = S_0 \times S_1 \times ... \times S_n$ is the finite set of environmental states. S_0 represents the full state of the environment and S_1 to S_n represent the local state representation for each agent in the system.
- $\mathrm{U} = \mathcal{A}_1 \times ... \times \mathcal{A}_n$ is the joint action space for all the agents, where $\mathcal{A}_k$ is the action space of agent k.
- $\mathcal{P} : \mathcal{S} \times \mathrm{U} \times \mathcal{S} \to [0,1]$ is a stochastic transition probability function that now depends on the actions of all agents.
- $\mathcal{R}_k : \mathcal{S} \times \mathrm{U} \to \mathbb{R}$ is the reward function of agent k that is now also affected by the actions of other agents.
- $\gamma \in [0,1]$ is the discount factor.

Since the reward function is individual to each agent and it now depends on the joint action taken that considers all agents, it is generally impossible to maximize the reward for all agents simultaneously.

To deal with this problem agents can learn as in the MDP framework by ignoring the other agents [5], try to reach an equilibrium policy [6] or maximize a common reward [7]. As mentioned before, by ignoring other agents, the actions of one agent might hinder the performance of another agent and no coordination is achieved.

Inspired by the area of Game Theory, some approaches try to find an equilibrium to solve this problem. For example, if each agent employs a strategy that calculates the best actions to take given the policy of other agents, they are playing according to a Nash equilibrium. This method can be directly applied in Q-Learning [8], where the Q-value update relies on the Nash equilibrium.

Another way to solve the problem is to work towards a common goal. If all agents have the same reward function $\mathcal{R}$ ($\mathcal{R}_1 = ... = \mathcal{R}_n$), at each time step, the joint action $u = (a_1, ..., a_n)$ results in a payoff function $\mathcal{R}(u)$. The coordination problem is now to find the optimal joint action u^* that maximizes $\mathcal{R}$, i.e., $u^* = \arg\max_a \sum_{t=0}^{\infty} \gamma^t \mathcal{R}(a_t)$

By enumerating all possible joint actions, a central agent can learn the best actions for every other agent but this leads to an exponential explosion in the state-action space that grows with the number of agents. Lauer et al. presented a Distributed Q-Learning method [9], where each agent only observes their own actions but this approach is limited to deterministic transition probability functions.

All of the previous approaches have performance issues when scaling the problem to a large number of agents. To deal with this, some approaches rely on more strict assumptions to build models that can handle a large number of agents that can coordinate to reach an objective.

2.3.1 Variable elimination coordination graphs

In many problem domains, such as the traffic control problem, the payoff function $\mathcal{R}(u)$ is sparse since at every time step each agent only interacts with a few other agents. This locality was exploited in [10] by the use of Coordination Graphs (CG) that decompose the global payoff function $\mathcal{R}(u)$ into a combination of local payoff functions. For example in a 4x1 arterial network with 4 agents, the payoff function could be decomposed as:

$$\mathcal{R}(u) = f_{12}(a_1, a_2) + f_{23}(a_2, a_3) + f_{34}(a_3, a_4) \tag{5}$$

where the functions f_{ij} represent the local payoff between the agent i and agent j. These functions can be arranged in a graph where each node represents an agent and each link represents a coordination dependency between two agents, hence the Coordination Graphs.

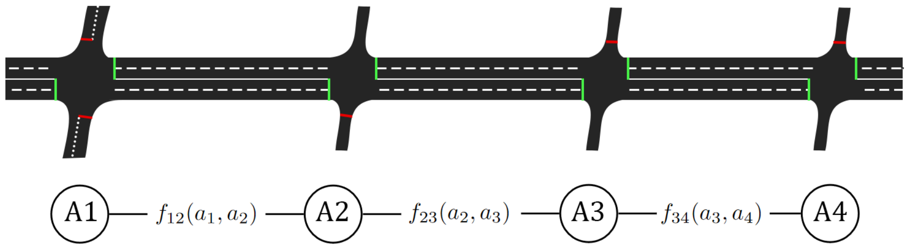


Fig. 3. Coordination graph in an arterial network.

After building this graph, the initial approach to solve the problem is to apply a Variable Elimination (VE) algorithm that selects one agent, optimizes all possible combinations of payouts that involve its neighbors and then eliminates it from the graph. This process is repeated until only one agent is left such that a final conditional strategy is formed. Then, a second iteration is performed in reverse order to select the action for each agent that complies with the chosen strategy in the first iteration. The VE algorithm result does not depend in the order the agents are eliminated and always produces the optimal joint action. However, the execution time depends on this order and in the worst case, the execution time grows exponentially with the number of agents. Thus, for real-time multi-agent systems such as the traffic control problem, this algorithm is not appropriate.

2.3.2 Max-Plus algorithm

To solve the performance issues in real-time systems with VE, an approximate algorithm such as Max-Plus [11], [12] can be used. This algorithm is used to compute the maximum a posteriori (MAP) setting in an undirected graph. In the multi-agent RL problem, computing the MAP is identical to computing the optimal joint action in a CG as shown in [13].

Assuming a CG with V agents and E coordination edges, the global payoff function can be defined as:

$$\mathcal{R}(a) = \sum_{i \in V} f_i(a_i) + \sum_{(i,j) \in E} f_{ij}(a_i, a_j) \tag{6}$$

where $f_i(a_i)$ represents a payoff function that is local to agent i, thus only depending on agent i actions. The goal of the max-plus algorithm is the same as the previous: find the optimal joint action u^* that maximizes the global payoff function. In this algorithm, every agent sends messages to its neighboring agents that map a real value to an action taken by the neighboring agent. The messages are defined as:

$$\mu_{ij}(a_j) = max_{a_i}\left(f_i(a_i) + f_{ij}(a_i, a_j) + \sum_{k\in\Gamma(i)\setminus j} \mu_{ki}(a_i)\right) + c_{ij} \tag{7}$$

where $\Gamma(i) \setminus j$ represents the neighbors of agent i except agent j, that is receiving the message, and c_{ij} is a normalization factor. This message represents an approximation of the maximum reward agent i can receive for a certain action of agent j. After a certain time exchanging messages, the algorithm will converge. This method is much faster than VE and can even be implemented in a distributed fashion, further improving the computation time.

2.3.3 Sparse-interaction approaches

There are other assumptions that can be made to the coordination solutions, for example, in some problem domains, the agents do not need to coordinate all the time. In other words, there are states where the agents need to coordinate and other states where they can act like single-agent systems since coordination is not required, this further improves the performance of the solution. These self-sufficiencies that only occur in certain states are labeled sparse interactions [14].

In the Utile Coordination algorithm [15], the agents only interact in certain states. These interactions are defined by coordination graphs and are learned online by using statistical information about the rewards of the agents. This approach reduces the size of the joint-action space but since actions are selected with a full view of the joint-action space, even in states where local information would be enough, the state-space is still exponential with the number of agents.

Spaan and Melo [16] defined interaction-driven Markov games (IDGM), a multi-agent model that contains a list of all the states where interaction between agents should occur. Later in [17], Melo and Veloso propose an algorithm where agents learn the states in which these interactions occur. In this model, every agent gains a new action, COORDINATE, that performs a perception step that is used to decided whether coordination is necessary or not. By penalizing failures in coordination, the agents learn to use this new action and learn the interaction states.

De Hauwer [18] describes the 2Observe, Coordinating Q-learning and Future Coordinating Q-learning algorithms that extend the previous models in different scenarios. The 2Observe algorithm replaces the list of coordination states/graphs with an Interaction function that receives the local state of the agent and returns a set of other states that the agent needs to worry about. In a non coordinating state, this set only contains the state of the own agent, while in a coordinating state this set may contain the state of other agents. In the mentioned algorithm, each agent has a set of generalized learning automata (GLA) that learn the interaction function. GLA's are associative RL units that are able to map a given input to an output without the need to store a value for each state-action pair.

In Coordinating Q-learning (CQ-learning), the agents identify when they are experiencing influence from other agents by the reward signal they receive. The algorithm stores the last N rewards it received for a certain state-action pair and performs a statistical test (one-sample Student t-test) with a baseline model to determine whether the agent is being influenced by others. The baseline model can be a single-agent model learned beforehand for each agent or can be provided to the agents. After identifying that it is being influenced, the local state is augmented with the state of the influencing agents so that they can coordinate when in that local state.

A modified version of the CQ-learning algorithm is also defined where instead of giving the baseline model before the algorithm starts, this model is learned by the rewards collected in the initial learning stage, since these are assumed to be good estimates of the rewards the agents would receive if they were alone in the environment.

CQ-learning is limited to domains where the influence of other agents is reflected in the immediate rewards that the agents receive, thus it is not suitable for delayed coordination problems.

Future Coordinating Q-learning (FCQ-learning) is similar to CQ-learning since it still uses statistical tests to determine if there is influence. It then augments the influenced states but it uses the Q-values instead of immediate rewards. Since the Q-values contain an approximation of the future delayed rewards, the influence of another agent can be detected a few time steps before the immediate reward is received.

Given that this work is concerned with the traffic control problem that has real-time dynamics, where traffic controllers need to make decisions in a short amount of time, the Max-plus algorithm is better suited to build RL traffic controllers. Regarding the sparse interactions between agents, the FCQ-learning algorithm is the most adequate for this problem since the actions of each agent only have effect on other agents a few time-steps later.

3 Related work

Traffic signal controllers have evolved over the years to better respond to the growing traffic flow by making use of the technological advancements in computation and the increasing data collected from sensors. This Section explores previous approaches that also try to solve traffic control problems.

3.1 Fixed-timing methods

Traffic signal controllers with fixed timings normally define different cycle profiles and alternate between them depending on the time of day on an attempt to deal with the different traffic flows that are usually observed.

Some of these methods are defined by mathematical models that use derivative calculus, linear programming, and other optimization algorithms:

- Webster [19] is a closed form method that calculates the cycle length and phase split that minimizes the travel time for a single intersection, assuming the traffic flow is uniform during a certain time period. This approach does not consider timing offsets for nearby traffic signals which can degrade the efficiency of transportation when considering a more complex environment.
- GreenWave [20] is a method that implements coordination by optimizing the timing offsets of nearby intersections to minimize the number of stops for vehicles moving in a certain direction.
- Maxband [21] also optimizes timing offsets by minimizing stops but now considers a two-way environment. It uses linear programming to achieve the maximal bandwidth in both ways for a set of arterials.

Other methods use traffic simulators to build the traffic model. For example, Rouphail et al. [22] applied a genetic algorithm using the CORSIM simulator to minimize link delay and queue time in a 9 intersection network but the results were limited by the slow convergence of the GA algorithm.

3.2 Adaptive systems

Later traffic controllers started using models that used sensor data to optimize different traffic metrics to better adapt to the changes in traffic flow:

- Max-pressure [23] aims to balance the queue length of neighboring intersections. The pressure of a phase is defined as the difference between the queue length of incoming and outgoing lanes. It is shown that Max-pressure maximizes the throughput of the system if the pressure of the phases is minimized.
- Scats [24] iteratively selects the next signal plan, from a set of pre-defined plans, depending on the current traffic conditions and a pre-defined performance measure. The model infers the performance of all plans before each cycle and then selects the plan that has better performance.
- Tubaishat et al [25] use the wireless sensor network to create a decentralized system that improves the localized flow and coordination between neighbor traffic lights.
- RHODES [26] is a hierarchical system that predicts the traffic load on each link and allocates phase time according to the predictions.
- Liu et al. [27] developed a controller that identifies upstream and downstream vehicles, in intervals of 15 minutes, to measure their delay and then choosing a signal timing plan that minimizes it.
- Tan et al. [28] developed a traffic controller that senses the number of incoming vehicles and uses fuzzy logic to determine the green time of a single intersection.
- Lee et al. [29] also used a fuzzy logic controller but in multiple intersections. The controller takes decisions based on vehicle data of previous and following junctions.

These systems generally outperform fixed-timing controllers but were tested in very simple scenarios with a single intersection or with very restricted traffic assumptions, like uniform traffic flow and cannot adapt well to real world city-level traffic.

These systems are not well suited for the present work since we will be dealing with large network scenarios that have a high number of traffic lights that need to coordinate to effectively deal with the current traffic flow.

3.3 Reinforcement learning

Earlier models were limited by simplistic simulations and lack of computation power, but with the latest growths in these areas, highly complex simulators emerged and a became popular tool to model traffic control models.

Recently, reinforcement learning has become popular to build traffic signal controllers, since an agent can learn traffic control policies by interacting with the environment, without having a pre-defined model of the system.

The reinforcement learning framework fits naturally into the traffic light controller problem since we can define the traffic controller as the agent, the traffic data as the state representation and the phase controls as the agent's actions.

Different learning models have been explored to build traffic signal controllers. However, it is hard to compare the proposed solutions and results since the problem definition varies greatly between the literature.

This Section provides an overview of the literature that adopted a reinforcement learning approach to solve the traffic control problem. Section 3.3.1 explores the MDP definitions used in the literature, namely the states, actions and rewards that are most used. Section 3.3.2 explores the classic, single-agent models used in the literature. Section 3.3.3 extends the previous models to allow for multi-agent settings and Section 3.3.4 examines the different methods based on deep learning that were applied to the traffic control problem.

3.3.1 Modeling

Based on the MDP framework defined in Section 2.2, the most used representations for the states $\mathcal{S}$, actions $\mathcal{A}$ and rewards $\mathcal{R}$ in the literature are described in this Section.

In the context of the traffic control problem, the environment is the network of roadways that approach the intersections and the vehicles that traverse these roadways. Since the traffic problem has a complex and stochastic environment, most works rely on simulators that accommodate the RL framework. Even though in these simulators the environment is fully observable, using all of the data available might prove difficult when trying to apply the model to the real world. Thus, most works use state representations that are derived from data that real sensors would capture in a typical urban roadway.

The definition of the environment state in the literature is a combination of one or more of the following parameters:

- Queue length: the total number of waiting vehicles in a certain lane or a set of lanes. In some works, the number of approaching vehicles is also considered for this metric.
- Delay: the time elapsed since a vehicle has stopped in an intersection. A cumulative sum of the delay in all intersections visited by a vehicle is also used in scenarios with multiple connected intersections.
- Volume: the total number of vehicles, moving or not, in a certain lane or a set of lanes.
- Vehicle position: the position of each vehicle in the network, usually implemented with a boolean matrix, where each cell represents a portion of of the network and a value of 1 represents that a vehicle is in that location at a certain time.
- Vehicle Speed: the speed of each vehicle. Since this value can be influenced by the speed limit of each lane, this value is usually normalized by the speed limits.
- Saturation: The ratio between the current traffic flow and the maximum traffic flow in an intersection.
- Phase: information about the current phase. For example, the index of the current phase can be used in systems that have pre-defined phases or the time passed since the signal turned red to prevent a red signal from being active too long, promoting fairness between other lanes.

These state definitions can be simpler or more complex depending on the number and complexity of the metrics used. A more complex state representation is built in hopes to better represent reality, but may lead to very big state spaces that can hinder the training performance, without actually boosting the effectiveness of the model. It should be noted that some of these values are harder to obtain than others. For example, vehicle speed and position might be difficult to accurately obtain from sensors positioned in the roadway.

In the traffic control problem, the set of actions, $\mathcal{A}$, is the set of phase controls of the traffic controller. The phase control actions that are available to the controller depend on the assumptions of the model used. Specifically, if the controller needs to select phases from a pre-defined set of valid phases or can choose them freely; if it needs to follow the order of a pre-defined phase plan or it learns the best order; if the phase cycle time is fixed or variable.

Examples of actions used in the literature are:

- Phase duration: the agent chooses the current phase duration, following a pre-defined phase plan.
- Phase ratio: given a fixed cycle time, the agent defines the phase split ratio, usually taken from a set of pre-defined split ratios.
- Keep or change phase: at every decision point, the agent decides to either extend the current phase or change to next phase, following a pre-defined phase plan.

- Choose next phase: the agent decides what is the next phase, from a set of pre-defined phases. This way the agent builds the phase plan, without explicitly following a cycle.

Regarding the reward function, $\mathcal{R}$, in the traffic control problem the objective is, ultimately, to minimize the travel time of all vehicles in the network. However, this metric can be hard to obtain for every vehicle since most times the vehicle destination is unknown.

Other metrics that can be more easily measured after each action are used in the hope that minimizing them also minimizes the travel time of all vehicles.

Some of the metrics used in the literature are the following:

- Queue length: the sum of the number of waiting vehicles in a set of lanes or a phase.
- Delay: the cumulative delay (as defined in the state representation) of all vehicles in a set of lanes or phase.
- Volume: the total number of vehicles, moving or not, in a certain lane.
- Speed: the average speed of vehicles in a set of lanes or phase, normalised by the speed limit.
- Throughput: the number of vehicles that traverse one or more intersections in a period of time, for example, in the last phase.

The reward function is usually defined as a weighted sum of one or more of the previous metrics, this approach can be tricky since small changes in each weight can greatly impact the results of the model and there is no straightforward way of tuning these weights, thus, these are generally tuned empirically.

3.3.2 Classic reinforcement learning

The first distinction in the different reinforcement learning methods is if the transition probability function, $\mathcal{P}$, needs to be learned or not.

In model-based methods the agent learns a transition model that estimates the probability of moving between states given the possible actions and then computes the rewards of each transition. Then, using methods based on dynamic programming, it estimates the value function and makes decisions based on this estimation.

While model-based methods need to learn $\mathcal{P}$ and $\mathcal{R}$, model-free methods bypass this step and learn the value function or policy by interacting with the environment and observing the rewards directly.

In the context of the traffic control problem, learning the transition probability function means modeling the environment, such that metrics like vehicle speed, position and acceleration can be predicted.

Wiering [1] used a model-based approach in a multi-agent model that operates in a 6 intersection network where each intersection is controlled by an agent. Each controller receives the discretized position and destination of each car in the approaching lanes, leading to 2^{78} possible traffic situations. Even though the defined RL-controllers outperform more simple controllers, like fixed time and Longest Queue First (LQF)[1], this model assumes that each car can communicate with each traffic light controller, which is currently unfeasible. The network is also simplified since every street has the same number of lanes, resulting in an unrealistic homogeneous traffic pattern. The author also mentions the possibility of having smarter driving policies that avoid congested intersections, assuming the previous communication is made possible.

The previous work was extended in [30] by adding the direction of each car to the state and tested on a bigger network with 36 roads and 15 intersections. The RL controllers outperformed fixed controllers and co-learning between cars and traffic lights minimized waiting time by 50%.

Some works [1], [31]–[33] have tried model-based approaches but most of the research community adopts model-free methods since it is difficult to fully model $\mathcal{P}$, given the natural unpredictable behavior of human drivers.

Most works using model-free methods relied on algorithms like Q-learning and SARSA to learn a optimal traffic control policy. Thorpe at al. [34] created a model-free system using SARSA and compared the performance of 3 different state representations, namely, the volume and the presence of absence of vehicles in each section of the network, by dividing each lane in equal and unequal distance sections. The RL model outperformed fixed time and greatest volume controllers, independently of the state representation used, and the state with unequal distance sections outperformed the other 2 state representations.

Zhou et al. [35] compared a SARSA model with a fixed time approach. The authors used the saturation ratio between current and maximum flow and vehicle speed, both normalized in 7 discrete values for a total of 49 states. The SARSA model had almost half travel time when compared to an offline model when the traffic flow was low but had negligible difference in high traffic flow.

Tantawy et al. [36] compared different learning methods, state, action and reward representations and action selection methods. For a single intersection, an approach based on TD(λ) with eligibility traces outperformed Q-learning and SARSA. RL-controllers generally outperformed pre-timed controllers regardless of the state representation. When the arrival rate is uniform, the best approach was to follow a pre-defined phase plan but when the arrival rate varies, letting the controller choose the order and duration of each phase showed better results. For action selection mechanisms, a mix of ε-greedy and softmax had better online performance and faster convergence when compared to both ε-greedy and softmax alone.

Touhbi et al. [37] extended the network used by Tantawy et al. [36] with different traffic volumes and a modified state definition, and compared different reward functions. For low traffic flow, the reward definition is irrelevant for the controller performance. For high traffic flow a reward based on the cumulative delay outperformed rewards based on queue length and throughput.

Earlier reinforcement learning based controllers are applied to a single intersection since the state space grows exponentially with the number of intersections

[1] A policy that sets green phases to approaches with the longest queue first.

that are controlled. Given that single intersection models are oversimplified and cannot extrapolate to city level traffic, other works tried to apply reinforcement learning to multiple traffic junctions by building multi-agent models.

3.3.3 Multi-agent reinforcement learning

In a multi-agent setting, each agent controls one intersection in a traffic network with several intersections. This way, the explosion of the state space is minimized by making each agent operate on a small partition of the environment. In a non-collaborative approach, each agent tries to maximize a certain reward, such as queue length or cumulative delay, of its own intersection by using the state that represents that intersection.

Since the actions of one agent can affect other agents in nearby intersections, having isolated self-interested agents that only try to maximize the gain in their own intersection may lead to better local performance for some agents but worse global performance when dealing with large networks. Thus, some form of collaboration or information sharing between agents is necessary.

The naive approach would be to simply add information about all other intersections to the state space. This leads to an exponential growth that increases with the number of intersections and is unfeasible in larger networks.

Thus, the main challenge in the multi-agent setting is to implement coordination and information sharing between the agents while maintaining a state-space with manageable size.

Camponogara et al. [38] created a multi-agent system based on Q-learning and modeled as a distributed stochastic game. The authors applied the system in a simple network with 2 connected intersections and compared it with a random and Longest Queue First policy. The proposed multi-agent model had significant performance gains over the other two policies. However, the agents did not collaborate and the proposed scenario was very simplistic.

Steingröver et al. [39] extended the model-based system proposed by Wiering [1] by introducing communication between agents by adding a bit to the state representation that represents the traffic congestion in nearby intersections and by using this congestion information when estimating the optimal action.

This model was also extended by Iša et al. [40] by adding another bit to the state space that represents if a following lane is obstructed or not. The main problem in the previous approaches is the exponential increase in state-space. For example, the space of the model of Iša et al. [40], that adds bits for congestion and obstruction is four times larger than the original model proposed Wiering [1].

Kuyer et al. [41] used the Green Light District (GLD) simulator to design a vehicle-based model like the model-based approach created by Wiering in [1]. The system achieved coordination by using the Max-Plus algorithm, explained in Section 2.3.2. This algorithm deals with the dimensions of the problem by exploiting the locality of the agents, where an agent only communicates with other agents that are close to it. The model was compared to the original Wiering model [1] and to the extension made by Steingröver [39] that adds a congestion bit to the state space. The devised model outperformed the others in both small (3 to 4 intersections) and big (15 intersections) networks.

El-Tantawy et al. [2] created a coordinated multi-agent model based on Q-learning. They used queue length for the state representation, cumulative delay for the reward function and the Max-Plus algorithm to coordinate the agents. The authors used the locality of the agents and modular Q-learning, described in [42], where the state space is separated in different sections. The model outperformed fixed time controllers and an uncoordinated Q-learning model in a five intersection network.

The authors later extended this model [43] by testing the system in a 59 intersection network, simulated from a real roadway in Downtown Toronto, with real input data. The model outperformed the uncoordinated Q-learning model and the existing controllers in the real network.

Collaborative multi-agent systems are a way to handle the curse of dimensionality when considering complex traffic networks, and show better performance when compared to fixed-timing, single RL agents and non-collaborative multi-agent RL models. However, most works rely on either directly adding information about other agents to the state representation, which usually leads to a state-space explosion, or take advantage of coordination graph approaches, like the Max-plus algorithm, that exploits the space locality of the agents. The actual effectiveness of these coordination methods is hard to grasp since each work defines a different set of state-action representations and testing scenarios.

All of the previously mentioned works relied on tabular methods that use discrete representations, such as queue length and cumulative delay. This leads to the loss of information about the traffic environment and can potentially lead to suboptimal control policies.

The next step was to apply methods that used a continuous state representations that, although having fewer convergence guarantees, had potential to outperform the classic tabular methods, given the extra information they convey to the traffic controller.

Some works attempted to apply these function approximation methods using linear approximators, such as radial basis functions and tile coding [44], but most works focused on non-linear approximators with the use of deep learning methods.

3.3.4 Deep reinforcement learning

Deep learning is a sub-field of machine learning that studies neural networks with many layers, inspired by information-processing in biological systems.

Deep reinforcement learning uses deep neural networks as function approximators in a reinforcement learning model. For example, the Deep Q-Network (DQN) algorithm is a variation of Q-learning that uses a deep neural network as a function approximator for the Q-function. This algorithm has been popularized due to its impressive performance when playing Atari games [45].

Li et al [46] applied a Deep Q-Network with stacked auto-encoders to a single intersection using the Paramics [47] micro-simulator. The model uses the queue lengths of each lane as input, it can maintain the current phase or switch to the next and the reward is computed as the difference between flows in each direction. The model outperformed traditional reinforcement learning approaches but the network used was an oversimplified 2 phase scenario where every direction has only 2 lanes and no turns are allowed at all.

Arel et al [48] proposed a continuous RL algorithm that studied a five intersection traffic network and assigned an agent to each intersection, distinguishing between the 4 outermost agents that run a Longest Queue First (LQF) algorithm and a central agent that uses a DQN algorithm to handle the large state space. The goal of the agents is to minimize the average delay, the congestion and intersection blocking. Each outermost agent senses and controls local variables while also communicating their current traffic conditions to the central agent. When compared to a simple LQF model, the multi-agent model had lower delay times and less cross blocking at high arrival rates. In this model, only the center agent uses information about the nearby agents, instead of a full collaboration scheme. The proposed solution also relies heavily on the geometry of the proposed ring-like scenario, which may be hard to extrapolate to more complex networks with different geometries.

Van der Pol [49] applied a deep learning approach to single and multi-agent settings. The learning agent uses a Deep Q-Network (DQN) algorithm with a binary matrix as input that represents if a vehicle is present in a certain location or not. For a single intersection network, the DQN agent had better stability and performance when compared to a baseline agent using linear approximation. The author then compared the use of the architectures made to play Atari games [45] in the traffic control problem, namely batch normalization, prioritized experience replay and double Q-learning. The performance and stability of each method was tested. Apart from batch normalization, all methods had an improvement in stability and performance. A fine-tuned DQN agent was then created from the previous experiments and used in a multi-agent scenario in increasingly complex networks, applying the Max-Plus algorithm with transfer learning, a method where the local joint function between agents is learned in a smaller problem, a 2x1 intersection network, and is then used as initialization for the local joint function for harder problems (a 3x1 and 2x2 network grids). The DQN agent with transfer learning had better performance than the vehicle-based approach introduced by Wiering [1] and the Kuyer's extention [41] that used the base Max-Plus algorithm in Wiering's model.

Genders et al. [50] extended the state representation of Thorpe [34] with information about the current phase and considering not only the position but also the speed of vehicles in the network. The authors suggest that using this highly dimensional state representation allows the traffic controller to create more efficient policies given that there is more information about the traffic environment. The model proposed a single agent using a deep convolution network tested in a single cross-shaped intersection using the SUMO simulator [51]. It outperformed a shallow neural network that received the queue length and current phase as inputs, obtaining up to 82% reduction in cumulative delay.

Casas [52] explored increasingly complex scenarios by applying a single learning agent using a Deep Deterministic Policy Gradient (DDPG) algorithm in a actor-critic model. The states and rewards were based on the speed data collected by detectors placed in the road and the actions were to set the duration for each phase. This model was tested in 3 different scenarios against a random model and a multi-agent tabular Q-learning approach with no collaboration. For smaller networks, DDPG had the same performance but better stability than Q-learning but for medium to large networks both algorithms diverged and were on par performance wise. Even though it was shown that DDPG can be applied to the traffic problem, the chosen reward did not favor larger arterials from smaller roads and it applied the same weights to every detector, independently of their position and importance to the state and reward definitions.

Most of the previous works make very simplistic assumptions on the traffic control problem. For example, the simulated environment is fully observable and stationary, and real-world factors like weather, pedestrian behavior, illegal parking, noisy sensor data and vehicle accidents or other roadway obstructions are not modeled.

Aslani et al. [44] applied a multi-agent actor-critic approach, comparing a discrete and continuous version of the model using the AIMSUN simulator[2]. The authors also compared and modeled real-world traffic disruptions like impatient pedestrians, illegal parking, noisy states and vehicle accidents. The chosen state representation was the current phase index and the number of waiting and approaching vehicles leading to each intersection. The work compared fixed time, actuated, discrete state RL and variations of continuous state RL, such as tile coding and radial basis functions (RBF). These different approaches were tested in a scenario with no traffic disruptions, in scenarios exploring each one of the previously mentioned disruptions and a final scenario where all traffic disruption were explored. Overall, the continuous RL model outperformed all other tested models in all scenarios, for example, the continuous model had a 59% improvement in travel time when compared to a fixed time controller in the last scenario. The authors also compared the previously mentioned model against discrete state Q-learning(λ) and Bayesian Q-learning in a simplified network where the continuous RL model also outperformed these two classic RL models.

Earlier works that used multi-agent reinforcement learning to solve the traffic control problem relied on simple extensions of the state representation that cannot scale to larger networks.

Given that each intersection is only connected to a few other intersections in the traffic network, more recent works [41] [2] tackle the dimensionality problem by using the Max-plus algorithm (detailed in Section 2.3.2), exploiting the geometric locality of the agents.

[2] `http://www.AIMSUN.com`, Transporting Simulation Systems (TSS).

These approaches are usually compared to fixed time controllers and non-coordinated versions of the same system. However, it is hard to compare them between themselves since they use different state-action representations and scenarios. It is also worth noting that no work has explored the time locality introduced by sparse interactions, defined in Section 2.3.3.

The current work proposes to explore two multi-agent coordination mechanisms: coordination graphs and sparse interaction approaches to compare them to other baseline traffic controllers in order to understand how effective they are.

4 Solution proposal

In this Section we describe our proposed solution to the traffic control problem. We start by presenting the simulation environment in Section 4.1, followed by the evaluation metrics and methods used in Section 4.3 and Section 4.2, respectively.

4.1 Simulation environment

Traffic micro-simulators like Paramics [47], SUMO [51] and AIMSUN[3] have been used to evaluate current traffic controllers and prototype new ones. In the reinforcement learning context, they can be used to model the environment where the agent learns the traffic policy.

In contrast to traffic macro-simulators, that simulate traffic flow as a whole, traffic micro-simulators model each vehicle individually, i.e. each vehicle property, such as position, speed, acceleration, route and emission rates are simulated for each vehicle.

The environment in this work will be simulated using SUMO [51], an open-source micro-simulator. To train and later compare the RL models, a few scenarios depicting different types of road networks, taken from the city of Lisbon, will be considered:

- Arterial road: a 3x1 arterial road (Figure 4) with a 4 lane road that intersects 3 smaller roads.
- Grid of intersections: grid networks with increasing size (Figure 5 and 6).

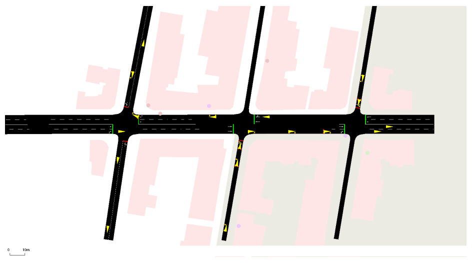


Fig. 4. The arterial road network.

[3] `http://www.AIMSUN.com`, Transporting Simulation Systems (TSS).

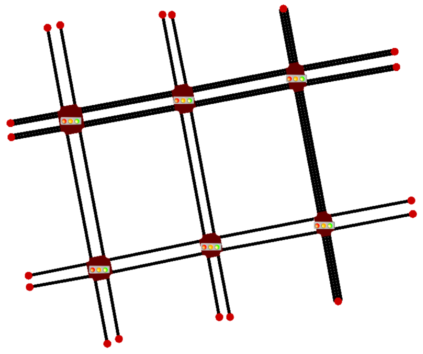


Fig. 5. Small grid network with 6 traffic controllers.

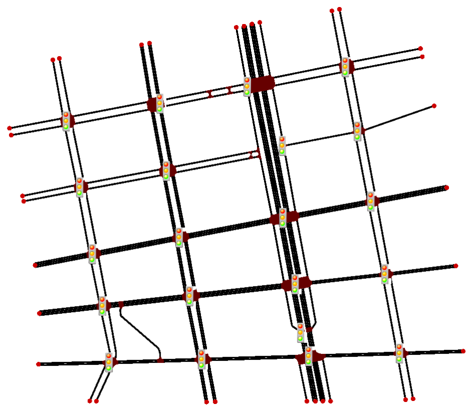


Fig. 6. City Network with 21 traffic controllers.

Real-world traffic networks usually deal with a high amount of traffic flow during peak hours, where most people go to or come from work. Outside of these peak hours there is a small amount of traffic flow. Thus, each scenario will also be tested using low and high traffic flow to understand the effectiveness of each traffic controller in these different situations.

4.2 Methods

The different state, action and reward definitions used in the literature were enumerated in Section 3.3.1. The choices are diverse and their performance is hard to compare. In this work, the performance of different state definitions, such as queue length, cumulative delay and volume will be compared. Different reward metrics, like queue length and cumulative delay will also be tested. In particular, we propose to compare:

– Coordination graphs: similar to Kuyer [41] and Tantawy [2], an algorithm based on the tabular Q-learning for multi-agent systems extended with the Max-Plus algorithm that ensures coordination between agents will be used.
– Sparse Interactions: An approach based on the concept of sparse interactions detailed in Section 2.3.3 will be used. Specifically, we will test the Future Coordinating Q-learning algorithm defined by De Hauwer [18]. Like the previous approach, different state and reward definitions will be tested. Sparse interactions exploit a structure of the problem distinct from coordination graphs. And while the latter have been extensively studied in the context of traffic control, the same cannot be said of approaches using sparse interactions. Our work will provide a comparison of the two, investigating their relative merits and disadvantages in the context of traffic control.

The two previous coordinated approaches will also be compared with two baseline models:

- Fixed time: A fixed time controller based on the Webster method [19].
- Uncoordinated Q-learning: An extension to the single agent tabular Q-learning, where agents select actions assuming the environment is stationary, thus, not having any type of coordination between themselves.

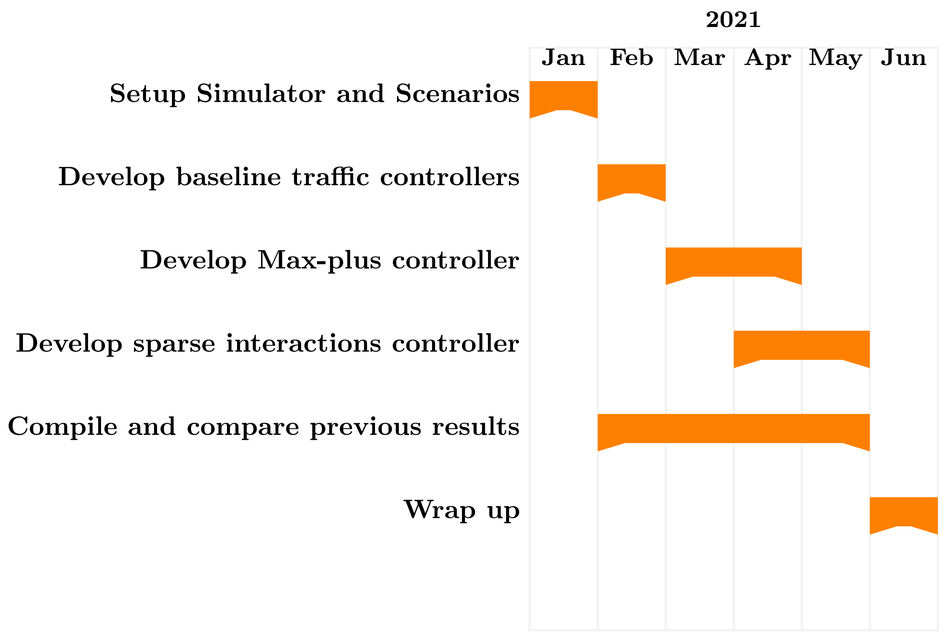
4.3 Evaluation

Given the discussion of the traffic control evaluation methods in Section 2.1.2, the performance of the proposed model will be evaluated by the following metrics:

- Travel time: The average travel time of all vehicles in the network, in other words, the average time since each vehicle entered and exited the network.
- Cumulative delay: As defined in Section 3.3.1, the time each vehicle as been in a waiting state since it entered the network.
- Number of stops: The average number of stops of each vehicle.

The multi-agent models will also be compared in terms of traffic performance and computational cost to a fixed time model using the Webster method and a multi-agent reinforcement learning model that does not consider any coordination between the agents.

5 Work Schedule



6 Conclusion

Due to the latest increase in demand for public and private transportation and the unchanging urgency of short travel times, smarter traffic controllers prove to have significant performance gains over the current traffic control schemes.

Smart traffic controllers based on coordinated multi-agent RL models are shown to be effective, specially when dealing with large traffic networks. Even though previous works have extensively studied approaches that exploit the space locality of the problem, using coordination graphs, the effectiveness of sparse interactions, i.e, approaches that exploit time locality in the traffic control problem is still unknown.

This work proposes to compare the two previously mentioned methods to understand if exploiting this time locality has any performance advantages over the classical methods that employ space locality, based on coordination graphs.

7 References

[1] M. Wiering, Multi-Agent Reinforcement Learning for Traffic Light Control. Jan. 1, 2000, p. 1158, 1151 pp.

[2] S. El-Tantawy and B. Abdulhai, “Multi-Agent Reinforcement Learning for Integrated Network of Adaptive Traffic Signal Controllers (MARLIN-ATSC),” in 2012 15th International IEEE Conference on Intelligent Transportation Systems, Sep. 2012, pp. 319–326. DOI: `10.1109/ITSC.2012.6338707`.

[3] R. S. Sutton and A. G. Barto, Reinforcement Learning: An Introduction. Bradford Books, Feb. 26, 1998, 322 pp.

[4] C. J. C. H. Watkins and King’s College (University of Cambridge), “Learning from delayed rewards,” 1989.

[5] M. Tan, “Multi-Agent Reinforcement Learning: Independent vs. Cooperative Agents,” in In Proceedings of the Tenth International Conference on Machine Learning, Morgan Kaufmann, 1993, pp. 330–337.

[6] Y. Hu, Y. Gao, and B. An, “Accelerating Multiagent Reinforcement Learning by Equilibrium Transfer,” IEEE Transactions on Cybernetics, vol. 45, no. 7, pp. 1289–1302, Jul. 2015, ISSN: 2168-2275. DOI: `10.1109/TCYB.2014.2349152`.

[7] L. Panait and S. Luke, “Cooperative Multi-Agent Learning: The State of the Art,” Autonomous Agents and Multi-Agent Systems, vol. 11, no. 3, pp. 387–434, Nov. 1, 2005, ISSN: 1573-7454. DOI: `10.1007/s10458-005-2631-2`. [Online]. Available: `https://doi.org/10.1007/s10458-005-2631-2` (visited on 12/21/2020).

[8] J. Hu and M. P. Wellman, “Nash q-learning for general-sum stochastic games,” The Journal of Machine Learning Research, vol. 4, pp. 1039–1069, null Dec. 1, 2003, ISSN: 1532-4435.

[9] M. Lauer and M. Riedmiller, “An Algorithm for Distributed Reinforcement Learning in Cooperative Multi-Agent Systems,” in In Proceedings of the Seventeenth International Conference on Machine Learning, Morgan Kaufmann, 2000, pp. 535–542.

[10] C. Guestrin, D. Koller, and R. Parr, “Multiagent Planning with Factored MDPs,” Advances in Neural Information Processing Systems, vol. 14, pp. 1523–1530, 2001. [Online]. Available: `https://proceedings.neurips.cc/paper/2001/hash/7af6266cc52234b5aa339b16695f7fc4-Abstract.html` (visited on 12/22/2020).

[11] J. R. Kok and N. Vlassis, “Collaborative Multiagent Reinforcement Learning by Payoff Propagation,” p. 40, 2006.

[12] ——, “Using the Max-Plus Algorithm for Multiagent Decision Making in Coordination Graphs,” RoboCup 2005: Robot Soccer World Cup IX, Lecture Notes in Computer Science, pp. 1–12, 2006. DOI: `10.1007/11780519_1`.

[13] N. Vlassis, R. Elhorst, and J. Kok, Anytime Algorithms for Multiagent Decision Making Using Coordination Graphs. Jan. 1, 2004, 957 vol.1, 953 pp. DOI: `10.1109/ICSMC.2004.1398426`.

[14] F. S. Melo and M. Veloso, “Decentralized MDPs with sparse interactions,” Artificial Intelligence, vol. 175, no. 11, pp. 1757–1789, Jul. 1, 2011, ISSN: 0004-3702. DOI: `10.1016/j.artint.2011.05.001`. [Online]. Available: `http://www.sciencedirect.com/science/article/pii/S0004370211000634` (visited on 12/26/2020).

[15] J. Kok, P. Hoen, B. Bakker, and N. Vlassis, Utile Coordination: Learning Interdependencies Among Cooperative Agents. Jan. 1, 2005.

[16] M. T. J. Spaan and F. S. Melo, “Interaction-driven Markov games for decentralized multiagent planning under uncertainty,” in Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems - Volume 1, ser. AAMAS ’08, Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, May 12, 2008, pp. 525–532, ISBN: 978-0-9817381-0-9.

[17] F. S. Melo and M. Veloso, “Learning of coordination: Exploiting sparse interactions in multiagent systems,” in Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems - Volume 2, ser. AAMAS ’09, Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, May 10, 2009, pp. 773–780, ISBN: 978-0-9817381-7-8.

[18] Y.-M. De Hauwer, “Sparse Interactions in Multi-Agent Reinforcement Learning,” in Uitgeverij VUBPRESS Brussels University Press, 2011, ISBN: 978 90 5487 920 6. [Online]. Available: `https://ai.vub.ac.be/wp-content/uploads/2019/12/Sparse-Interactions-in-Multi-Agent-Reinforcement-Learning.pdf` (visited on 12/26/2020).

[19] F. V. Webster and Road Research Laboratory, Traffic Signal Settings. London: H.M.S.O., 1958.

[20] R. P. Roess, E. S. Prassas, and W. R. McShane, Traffic Engineering. Pearson/Prentice Hall, 2004, 786 pp., ISBN: 978-0-13-191877-1. Google Books: akMaQAAACAAJ.

[21] J. Little, M. Kelson, and N. Gartner, "MAXBAND: A Program for Setting Signals on Arteries and Triangular Networks," Transportation Research Record Journal of the Transportation Research Board, vol. 795, pp. 40–46, Dec. 1, 1981.

[22] N. M. Rouphail, B. B. Park, and J. Sacks, "Direct Signal Timing Optimization: Strategy Development and Results," In XI Pan American Conference in Traffic and Transportation Engineering, 2000.

[23] P. Varaiya, "The Max-Pressure Controller for Arbitrary Networks of Signalized Intersections," 2013. DOI: 10.1007/978-1-4614-6243-9_2.

[24] P. R. Lowrie, Roads and Traffic Authority of New South Wales, and Traffic Control Section, SCATS, Sydney Co-Ordinated Adaptive Traffic System: A Traffic Responsive Method of Controlling Urban Traffic. Darlinghurst, NSW, Australia: Roads and Traffic Authority NSW, Traffic Control Section, 1990.

[25] M. Tubaishat, Y. Shang, and H. Shi, "Adaptive Traffic Light Control with Wireless Sensor Networks," 2007 4th Annual IEEE Consumer Communications and Networking Conference, CCNC 2007, Jan. 1, 2007. DOI: 10.1109/CCNC.2007.44.

[26] P. Mirchandani and L. Head, "A real-time traffic signal control system: Architecture, algorithms, and analysis," Transportation Research Part C: Emerging Technologies, vol. 9, no. 6, pp. 415–432, Dec. 2001, ISSN: 0968-090X. DOI: 10.1016/S0968-090X(00)00047-4. [Online]. Available: https://asu.pure.elsevier.com/en/publications/a-real-time-traffic-signal-control-system-architecture-algorithms (visited on 11/03/2020).

[27] H. Liu, J.-S. Oh, and W. Recker, "Adaptive Signal Control System with Online Performance Measure for a Single Intersection," Transportation Research Record, vol. 1811, no. 1, pp. 131–138, Jan. 1, 2002, ISSN: 0361-1981. DOI: 10.3141/1811-16. [Online]. Available: https://doi.org/10.3141/1811-16 (visited on 11/03/2020).

[28] K. Khiang, M. Khalid, and R. Yusof, "Intelligent Traffic Lights Control By Fuzzy Logic," Malaysian Journal of Computer Science, vol. 9, pp. 29–35, Jan. 1, 1997.

[29] J.-h. Lee, K.-m. Lee, K. Seong, C. Kim, and H. Lee-kwang, "Traffic Control Of Intersection Group Based On Fuzzy Logic," in In Proceedings of the 6th International Fuzzy Systems Association World Congress, 1995, pp. 465–468.

[30] M. Wiering, J. Veenen, J. Vreeken, and A. Koopman, "Intelligent Traffic Light Control," Aug. 9, 2004.

[31] M. Wiering, J. Vreeken, J. Veenen, and A. Koopman, Simulation and Optimization of Traffic in a City. Jul. 14, 2004, p. 458, 453 pp., ISBN: 978-0-7803-8310-4. DOI: 10.1109/IVS.2004.1336426.

[32] A. Salkham and V. Cahill, "Soilse: A decentralized approach to optimization of fluctuating urban traffic using Reinforcement Learning," IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC, pp. 531–538, Sep. 1, 2010. DOI: 10.1109/ITSC.2010.5625145.

[33] M. Khamis and W. Gomaa, "Enhanced Multiagent Multi-Objective Reinforcement Learning for Urban Traffic Light Control," Proceedings - 2012 11th International Conference on Machine Learning and Applications, ICMLA 2012, vol. 1, p. 591, Dec. 1, 2012. DOI: 10.1109/ICMLA.2012.108.

[34] T. L. Thorpe and C. W. Anderson, "Traffic Light Control Using SARSA with Three State Representations," IBM Corporation, 1996.

[35] X. Zhou, F. Zhu, Q. Liu, Y. Fu, and W. Huang. (Jan. 23, 2014). "A Sarsa()-Based Control Model for Real-Time Traffic Light Coordination," The Scientific World Journal, [Online]. Available: https://www.hindawi.com/journals/tswj/2014/759097/ (visited on 11/09/2020).

[36] S. El-Tantawy, B. Abdulhai, and H. Abdelgawad, "Design of Reinforcement Learning Parameters for Seamless Application of Adaptive Traffic Signal Control," Journal of Intelligent Transportation Systems, vol. 18, Jun. 13, 2014. DOI: 10.1080/15472450.2013.810991.

[37] S. Touhbi, M. A. Babram, T. Nguyen-Huu, N. Marilleau, M. L. Hbid, C. Cambier, and S. Stinckwich, "Adaptive Traffic Signal Control : Exploring Reward Definition For Reinforcement Learning," Procedia Computer Science, 8th International Conference on Ambient Systems, Networks and Technologies, ANT-2017 and the 7th International Conference on Sustainable Energy Information Technology, SEIT 2017, 16-19 May 2017, Madeira, Portugal, vol. 109, pp. 513–520, Jan. 1, 2017, ISSN: 1877-0509. DOI: 10.1016/j.procs.2017.05.327. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1877050917309912 (visited on 11/09/2020).

[38] E. Camponogara and W. Kraus, "Distributed Learning Agents in Urban Traffic Control," vol. 2902, p. 335, Nov. 6, 2003. DOI: 10.1007/978-3-540-24580-3_38.

[39] M. Steingrover, R. Schouten, S. Peelen, E. Nijhuis, and B. Bakker, Reinforcement Learning of Traffic Light Controllers Adapting to Traffic Congestion. Jan. 1, 2005, p. 223, 216 pp.

[40] J. Isa, J. Kooij, R. Koppejan, and L. Kuijer, "Reinforcement Learning of Traffic Light Controllers Adapting to Accidents," p. 14, Februrary 2, 2006. DOI: 10.1.1.386.9994.

[41] L. Kuyer, S. Whiteson, B. Bakker, and N. Vlassis, Multiagent Reinforcement Learning for Urban Traffic Control Using Coordination Graphs. Sep. 15, 2008, p. 671, 656 pp. DOI: 10.1007/978-3-540-87479-9_61.

[42] N. Ono and K. Fukumoto, "Multi-agent Reinforcement Learning: A Modular Approach," p. 7, 1996. DOI: 10.1007/3-540-62934-3_39.

[43] S. El-Tantawy, B. Abdulhai, and H. Abdelgawad, "Multiagent Reinforcement Learning for Integrated Network of Adaptive Traffic Signal Controllers (MARLIN-ATSC): Methodology and Large-Scale Application on

Downtown Toronto," IEEE Transactions on Intelligent Transportation Systems, vol. 14, no. 3, pp. 1140–1150, Sep. 2013, ISSN: 1558-0016. DOI: 10.1109/TITS.2013.2255286.

[44] M. Aslani, M. S. Mesgari, and 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, vol. 85, pp. 732–752, Dec. 1, 2017, ISSN: 0968-090X. DOI: 10.1016/j.trc.2017.09.020. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0968090X17302681 (visited on 11/23/2020).

[45] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing Atari with Deep Reinforcement Learning," Dec. 19, 2013.

[46] L. Li, Y. Lv, and F. Wang, "Traffic signal timing via deep reinforcement learning," IEEE/CAA Journal of Automatica Sinica, vol. 3, no. 3, pp. 247–254, Jul. 2016, ISSN: 2329-9274. DOI: 10.1109/JAS.2016.7508798.

[47] G. D. B. Cameron and G. I. D. Duncan, "PARAMICS—Parallel microscopic simulation of road traffic," The Journal of Supercomputing, vol. 10, no. 1, pp. 25–53, Mar. 1, 1996, ISSN: 1573-0484. DOI: 10.1007/BF00128098. [Online]. Available: https://doi.org/10.1007/BF00128098 (visited on 01/06/2021).

[48] I. Arel, C. Liu, T. Urbanik, and A. Kohls, "Reinforcement learning-based multi-agent system for network traffic signal control," Intelligent Transport Systems, IET, vol. 4, pp. 128–135, Jul. 1, 2010. DOI: 10.1049/iet-its.2009.0070.

[49] E. van der Pol, "Deep Reinforcement Learning for Coordination in Traffic Light Control (MSc thesis)," Aug. 29, 2016.

[50] W. Genders and S. Razavi, "Using a Deep Reinforcement Learning Agent for Traffic Signal Control," Nov. 3, 2016.

[51] M. Behrisch, L. Bieker-Walz, J. Erdmann, and D. Krajzewicz, SUMO – Simulation of Urban MObility: An Overview. Oct. 1, 2011, vol. 2011, ISBN: 978-1-61208-169-4.

[52] N. Casas. (Aug. 2, 2017). "Deep Deterministic Policy Gradient for Urban Traffic Light Control." arXiv: 1703.09035 [cs], [Online]. Available: http://arxiv.org/abs/1703.09035 (visited on 11/23/2020).