NEXTRANS Project No. 072IY03

Agent-based Traffic Management and Reinforcement Learning in Congested Intersection Network

By
Juan C. Medina
PhD Candidate
University of Illinois at Urbana-Champaign
jcmedina@illinois.edu

and

Rahim F. Benekohal
Professor
University of Illinois at Urbana-Champaign
rbenekoh@illinois.edu

Final Report
August 23, 2012

TABLE OF CONTENTS

CHAPTER 1. INTRODUCTION

Traffic control systems have evolved from traditional pre-timed isolated signals to actuated and coordinated corridors, and more recently to more complex “adaptive” signal control systems. These improvements go along with increases in traffic demands and urban congestion, which makes necessary to use as much of the available roadway capacity in the most efficient and economical way. Even though most advanced traffic control signals can, react to changes in traffic, to certain degree, unexpected variations in demands, oversaturation, the occurrence of incidents, and adverse weather conditions, among others, may significantly impact the traffic network operation.

This study evaluates the performance of a traffic control system that diverges from traditional approaches. The proposed implementations make use of machine learning techniques, and more specifically of unsupervised learning through algorithms of reinforcement learning (RL), also called approximate dynamic programming (ADP) in some research communities (Gosavi, 2009). Potential benefits of this approach include signal systems that not only quickly respond to the actual conditions found in the field, but also learn about them and truly adapt through flexible cycle-free strategies. Moreover, these signal systems are decentralized, providing greater scalability and lower vulnerability at the network level.

The RL algorithms have been implemented using a commercially available microscopic traffic simulation (VISSIM) that allows to operate the traffic signals in real time through a communications port that is accessed in running time. Thus, the potential for these systems can be tested practically in any scenario that can be simulated. This report includes the performance of two particular algorithms (Q-learning, and ADP with post-decision state variable) building on a case for a single intersection, and then moving onto two different arterials, to finally arrive to a realistic network of 20 intersections.

A series of indicators or measures of performance are examined to determine how the algorithms behave in the proposed scenarios. The main focus of this study goes to oversaturated conditions, but some scenarios where the demand is lower than the capacity are also studied.

The remaining of the report is organized as follows. First, a review of current literature on agent-based approaches using reinforcement learning is provided. Then, Chapter 3 describes the methodology, the algorithms, as well as the implementation of real-time control in the simulation software. In Chapter 4, the results and analysis of the performance of the RL control is presented, as mentioned above, starting from a single intersection and moving to a mid-sized network. Lastly, Chapter 5 includes the conclusions and recommendations for future studies.

CHAPTER 2. BRIEF ACCOUNT OF PAST RESEARCH ON AGENT-BASED TRAFFIC CONTROL

Very broadly, today's advanced traffic signal systems could be grouped as traffic responsive and traffic adaptive systems, following a classification proposed in the Traffic Control Systems Handbook (Gordon et al, 2005). Traffic responsive systems make use of vehicle detectors to determine the best gradual changes in cycles, splits, and offsets, for intersections within a predetermined sub-area of a network. Well known examples in this category are the SCOOT and SCATS systems. On the other hand, adaptive systems have more flexibility in the signal parameters and they do not make use of predetermined signal timing settings for their operation. In addition to sensor information, they also use prediction models to estimate traffic arrivals at intersections and adjust the signal settings to optimize an objective function, such as delay. Examples of adaptive systems are RHODES and OPAC, which optimize an objective function for a specified rolling horizon (using traffic prediction models) and have pre-defined sub-areas (limited flexibility) in which the signals can be coordinated.

Alternative methods for real-time traffic signal control have been previously proposed based on developments from the field of machine learning. These strategies can solve stochastic optimization problems that are difficult to model (the expectation of the transition function is difficult to be computed), require sequential decision making, and have high dimensional decision and solution spaces, similar to the problem of finding optimal signal timings in a traffic network.

In the 1980s, long-acknowledged limitations in the application of exact dynamic programming methods to solve large stochastic optimization problems prompted the search for alternative strategies. Different research communities including those from the operations research and artificial intelligence started developing a series of algorithms to solve Bellman's optimality equation (at least approximately), finding near-optimal solutions for large scale problems. Among other methods, members of the artificial intelligence community proposed what is it known as a reinforcement learning (RL) approach by combining the concepts from classical DP, adaptive function approximations (Werbos, 1987) and learning methods (Barto el at, 1983).

Q-learning is one of such reinforcement learning strategies. After its initial publication (Watkins, 1989, 1992 – Watkins and Dayan, 1992), many studies have followed on the analysis of this and other algorithms based on similar principles. A good example is the analysis of reinforcement learning published by Sutton and Barto (1998) with their book "Reinforcement Learning: An Introduction", which covers the reinforcement learning problem, a series of methods for solving it (dynamic programming, Monte-Carlo, and temporal difference methods), extensions, and case studies. Similar learning algorithms include Sarsa and actor-critic methods, but the focus here will be given to Q-learning, mostly giving its off-policy nature of doing temporal difference control.

Q-learning has been the research topic of numerous practical applications, leading to enhancements in the algorithm and its learning structure. For example, a combination of Q-learning and principles of temporal difference learning (Sutton, 1988 – Tesauro, 1992) resulted in the Q(λ) algorithm (Watkins, 1989 – Peng and Williams, 1991) for non-deterministic Markov decision processes. In Q(λ) – which implements an eligibility trace, the updates are allowed not only for the last visited state-action pair but also for the preceding predictions. The eligibility is based on a factor that decreases exponentially over time (given that the discount factor for delayed rewards is lower than one and that lambda is greater than zero). Thus, the original version of the Q-learning algorithm is equivalent to a Q(0)-learning, and on the other end the traces can extend the full extent of the episodes when Q(1)-learning is used. In terms of its performance and robustness, the Q(λ) algorithm has shown improvements over the 1-step Q-learning (Pendrith, 1994 – Rummery and Nirajan, 1994 – Peng, 1993), and it is a viable option for the traffic control problem. Also, several other forms of Q-learning approaches have emerged with enhanced capabilities, such as W-learning (Humphrys, 1995, 1997), HQ-learning (Wiering, 1997), Fast Online Q(λ) (Wiering, 1998), and Bayesian Q-learning (Dearden, et al, 1998).

Thus, it could be said that the study and development of reinforcement learning has benefitted from a great number of approaches. The fields of classical dynamic programming, artificial intelligence (temporal difference), stochastic approximation (simulation), and function approximation have all contributed to reinforcement learning in one or other way (Gosavi, 2009).

On the other hand, approximate dynamic programming (under such name) evolved based on the same principles as reinforcement learning, but mostly from the perspective of the

operations research. Also, in some sense, advances shown above for reinforcement learning are also advances in the approximate dynamic programming (ADP) field. As it is pointed by Powell (2007), the initial steps in finding exact solutions for Markov Decision Processes date back to the work by Bellman (1957) and Bellman and Dreyfus (1959), and even back to Robbins and Monro (1951), but it was not until the 1990s that formal convergence of approximate methods was brought to light mainly in the books by Bertsekas and Tsitsiklis (1996) and Sutton and Barto (1998) (even though this last one is focused from a computer science point of view). These two books are arguably the most popular sources for ADP methods, and quite a few significant works have followed, including the book by Powell (2007) itself, which covers in great detail some of the most common algorithms, and particularly the use of the post-decision state variable (which is widely used in this research).

2.1 Traffic Control Using Learning Agents

Specifically for traffic signal control, the study of reinforcement learning dates back about 15 years ago. One of the first of such studies was completed by Thorpe (1997), using the RL algorithm SARSA to assign signal timings to different traffic control scenarios. Later, Wiering (2000) discussed a state representation based on road occupancy and mapping the individual position of vehicles over time, and Bakker (2005) later extended this representation using an additional bit of information from adjacent intersections. This allowed communication between agents, trying to improve the reward structure and ultimately the overall performance of the system.

Using a different approach, Bingham (1998, 2001) defined fuzzy rules to determine the best allocation of green times based on the number of vehicles that would receive the green and red indication. He presented a neural network to store the membership functions of the fuzzy rules, reducing memory requirements. It is noted that a Cerebellar Model Articulation Controller (CMAC) has also been used in the past to store the information learned (Abdulhai, 2003). Another application using fuzzy rules for traffic control was presented by Appl and Brauer (2000), where the controller selected one of the available signal plans based on traffic densities measured at the approaching links. Using a single intersection, their fuzzy controller outperformed learning from a controller with a prioritized sweeping strategy.

Choy et al. (2003) also used a multi-agent application for traffic control, but creating a hierarchical structure with three levels: intersection, zones, and regions. The three types of agents (at each level) made decisions based on fuzzy rules, updated their knowledge using a reinforcement learning algorithm, and encoded the stored information through a neural network. Agents selected a policy from a set of finite possible policies, where a policy determined shortening, increasing, or not changing green times. Experiments on a 25-intersection network showed improvements with the agents compared to fixed signal timings, mostly when traffic volumes were higher.

Campoganara and Kraus (2003) presented an application of Q-learning agents in a scenario of two intersections next to each other, showing that when both of those agents implemented the learning algorithm, the systems performed significantly better than when only one of none of them did. The comparison was made with a best-effort policy, where the approach with longer queue received the green indication. Also, Medina et al. (2010) used Q-learning to manage the traffic signals of a 5-intersection arterial and showed emergent coordination along the corridor in scenarios with variable demands. A similar work by Medina and Benekohal (2011) showed the performance of the Q-learning algorithm in a 2x3 and a 3x3 network with loads near capacity, and found better results than using pretimed signals and more balanced operation in two-lane roadways.

A study on the effects of non-stationary nature of traffic patterns using RL was proposed by De Oliveira et al. (2006b), who analyzed the performance of RL algorithms upon significant volume changes. They pointed out that RL may have difficulties to learn new traffic patterns, and that an extension of Q-learning using context detection (RL-CD) could result in improved performance.

Ritcher et al (2007) showed results from agents working independently using a policy-gradient strategy based on a natural actor-critic algorithm. Experiments using information from adjacent intersections resulted in emergent coordination, showing the potential benefits of communication, in this case, in terms of travel time. Xie (2007) and Zhang (2007), explored the use of a neuro-fuzzy actor-critic temporal difference agent for controlling a single intersection, and used a similar agent definition for arterial traffic control where the agents operated independently from each other. The state of the system was defined by fuzzy rules based on

queues, and the reward function included a linear combination of number of vehicles in queue, new vehicles joining queues, and vehicles waiting in red and receiving green. Results showed improved performance with the agents compared to pre-timed and actuated controllers, mostly in conditions with higher volumes and when the phase sequence was not fixed.

Note that most of the previous research using RL has been focused on agents controlling a single intersection, or a very limited number intersections interacting along an arterial or a network. Most of the efforts have been on the performance of the agents using very basic state representations, and no studies focusing on oversaturated conditions and preventing queue overflows have been conducted. Additional research exploring the explicit coordination of agents and group formation in different traffic control settings will be reviewed in the next subsection, and will provide an important basis for the coordination of agents proposed in this study.

Regarding the application of exact dynamic programming (DP), only a few attempts at solving the problem of optimal signal timings in a traffic network are found in the literature. This is not surprising because even though DP is an important tool to solve complex problems by breaking them down into simpler ones - and generating a sequence of optimal decisions by moving backward in time to find exact global solutions – it suffers from what is known as the curses of dimensionality. Solving Belman's optimality equation recursively can be computationally intractable, since it requires the computation of nested loops over the whole state space, the action space, and the expectation of a random variable. In addition, DP requires knowing the precise transition function and the dynamics of the system over time, which can also be a major restriction for some applications.

Thus, with these considerations, there is only limited literature for medium or large-sized problems exclusively using DP. The work of Robertson and Bretherton (1974) is cited as an example of using DP for traffic control applications at a single intersection, and the subsequent work of Gartner (1983) for using DP and a rolling horizon, also for the same application.

On the other hand, Approximate Dynamic Programming (ADP) has increased potential for large-scale problems. ADP uses an approximate value function that is updated as the system moves forward in time (as opposed to standard DP), thus ADP is an "any-time" algorithm and this gives it advantages for real-time applications. ADP can also effectively deal with stochastic

conditions by using post-decision variables, as it will be explained in more detail in the subsequent Section.

Despite the fact that ADP has been used extensively as an optimization technique in a variety of fields, the literature shows only a few studies in traffic signal control using this approach. Nonetheless, the wide application of ADP in other areas has shown that it can be a practical tool for real-world optimization problems, such as signal control in urban traffic networks. An example of an ADP application is a recent work for traffic control at a single intersection by Cai et al. (2009), who used ADP with two different learning techniques: temporal-difference reinforcement learning and perturbation learning. In their experiments, the delay was reduced from 13.95 vehicle-second per second (obtained with TRANSYT) to 8.64 vehicle-second per second (with ADP). In addition, a study by Teodorvic et al. (2006) combined dynamic programming with neural networks for a real-time traffic adaptive signal control, stating that the outcome of their algorithm was nearly equal to the best solution. Lastly, Hajbabaie, Medina, and Benekohal (2011) used approximate dynamic programming and compared it to genetic algorithms and TRANSYT7F in an oversaturated network of 20 intersections.

2.2 Explicit Coordination of Agents and Group Formation

Additional efforts have been conducted to incorporate explicit coordination to the behavior of groups of agents so that they can act together and form temporary coalitions. There is extensive research in this area for other applications other than traffic control, and most of the work has been originated from the artificial intelligent community. Given the focus of this study, review on this topic is centered on cooperative agents that share or exchange some information to achieve better system-wide performance, and where the communication is achieved in a completely decentralized way.

Communication between agents, without mediation from agents with higher hierarchies, may allow the formation of (temporary) groups that can improve the overall performance of the system. For the traffic control domain, it is of outmost importance to maintain acceptable operational levels in the whole network, since queue spillbacks and traffic breakdowns may extend to greater areas and ultimately collapse the system. For the particular case of traffic signal control, researchers have explored some mechanisms to communicate agents and improve

performance. Nunez et al. (2002) included a feature for heterogeneous agents to request advice from agents with better performance index, similar to supervised learning. Agents exchanged their state, the best action for such state (as a means of advice), as well as their performance index. The effects of the advice exchange were tested using a series of individual intersections (not along an arterial) in a very simple simulation, each with an agent that had a different learning algorithm. Results showed that the advice exchange was likely to improve performance and robustness, but ill advice was also said to be a problem hindering the learning process.

De Oliveira et al. (2006a) used a relationship graph as a support of the decision-making process. Related agents entered a mediation process to determine the best set of actions. Agents have priorities and the one with highest value will lead the mediation. Branch-and-bound was performed to find the best outcome of the sub-problem. The test was conducted on a 5x5 network in a very simple simulation environment provided by a generic tool for multi-agent systems (not a traffic-specific environment). Temporary group formation was achieved and resulted in improved performance in terms of a cost function, compared to pre-timed coordinated signals. The agents regrouped (through a new mediation) when traffic patterns changed, adapting to new conditions.

Kuyer (2008) also used coordination graphs and the max-plus algorithm to connect intersections close to each other. Networks having up to 15 intersections were tested, finding improved results compared to Wiering (1997) and Bakker (2005). Oliveira and Bazzan (2004, 2006, and 2007) have made significant contributions using approaches based on swarm intelligence, where agents behave like a social insect and the stimuli to select one phase or plan is given by a “pheromone” trail with an intensity related to the number and duration of vehicles in the link.

A different approach by Junges and Bazzan (2007) also studied a strategy using a distributed constraint optimization problem for networks of up to 9x9 intersections, but only for the task of changing the offset of the intersections given two different signal plans. A scenario without online capabilities to change the coordinated direction was compared another with the coordination scheme, showing improvements in the performance. However, for frequent action evaluations, and for bigger networks, the methodology may not be practical as the computation time increases exponentially with the number of agents.

A summary of past research using RL for traffic control is shown in Tables 2.1 and 2.2., where the state and the reward representation of the different approaches are described. The implementations presented in this report will be based on modifications and variations of previous work, with the addition of factors that may improve the system performance particularly in oversaturated conditions, including the explicit coordination of agents through the use of the max-plus algorithm.

Author	Algorithm	State and actions	Communication Between Agents	Application	Loads	Training
Thorpe	SARSA with eligibility traces	**State:** Number of vehicles (vehicles grouped in bins). **Actions:** unidimensional (direction to receive green)	No	4x4 network different loads	Mutiple (undersaturation)	On a single intersection, then use same training for network
Thorpe	SARSA with eligibility traces	**State:** Link occupation (link divided in equal segments). **Actions:** unidimensional (direction to receive green)	No	4x4 network different loads	Mutiple (undersaturation)	On a single intersection, then use same training for network
Thorpe	SARSA with eligibility traces	**State:** Link occupation (link divided in unequal segments). **Actions:** unidimensional (direction to receive green)	No	4x4 network different loads	Mutiple (undersaturation)	On a single intersection, then use same training for network
Thorpe	SARSA with eligibility traces	**State:** Number of vehicles (vehicles grouped in bins), current signal status. **Actions** were represented by a minimum phase duration (8 bins) and the direction which receives green	No	4x4 network different loads	Mutiple (undersaturation)	All intersections shared common Q-values
Appl and Brauer	Q-function approximated by fuzzy prioritized sweeping	**State:** Link density distribution for each direction. **Actions:** plan selection, total of three possible plans	No	Single intersection	Not described, likely undersaturation	Not described
Wiering	Model-based RL (with Q values)	**State:** Number of vehicles in links. **Actions:** signal states/phases (up to 6 per intersection). Car paths are selected based on the minimum Q-value to the destination. This is a car-based approach. It uses values for each car and a voting approach to select actions	Yes (shared knowledge or "tables" in some scenarios). Also, included a look-ahead feature	2x3 network	Multiple (undersaturation/likely oversat for 1 case)	Not described
Bingham	Neurofuzzy controller with RL (using GARIC, an approach based on ANN)	**State:** Vehicles in approaches with green, and those in approaches with red (these are the inputs to the ANN). **Actions:** values of green extension: zero, short, medium, and long. Fuzzy rules depend on how many extensions have already been granted	No	Single intersection	Multiple (undersaturation/likely oversat for 1 case)	Not described
Gieseler	Q-learning	**State:** Number of vehicles in each of the approaches and a boolean per direction indicating if neighbors have sent vehicles "q" seconds earlier, quere "q" is the # of veh in queue. **Actions:** one of 8 possible actions at a single intersection	Yes, boolean variable showing if the signal was green "q" seconds earlier. Also shred information of the rewards	3x3 network	Not described	Not described
Nunes, Oliveira	Heterogeneous (some agents use Q-learning, others hill climbing, simulated annealing, or evolutionary algorithms). Then, the learning process is RL + advice from peers	**State:** two cases: one is the ratio of vehicles in each link to the total number of vehicles in the intersection (4 dimentions), and the second is equal to the first plus an indication showing the time of the front vehicle in queue - this is the longest time a vehicle has been in the link (additional 4 dimentions). **Action:** percent of time within the cycle that green will be given to N-S direction (the other direction receives the complement	Yes (advice exchange): communicate state and the action that was taken by the advisor agent, the present and past score	Single intersection - each agent controls one intersection but they are not connected	Not described	Boltzman distribution used for action selection (T factor between 0.3 and 0.7); learning rate decreased over time for convergence
Abdulhai	Q-learning (CMAC to store Q-values)	**State:** Queue length of each of 4 approaches and phase duration. **Action:** Two possible phases with bounded cycle length	No, but recommended by sharing info on state and on rewards from a more global computation	Single intersection	Not described but variable over time (likely undersaturated)	E-greedy, Boltzman, and annealing (in separate experiments)
Choi et al	RL agents using fuzzy sets. Three hierarchies well defined	**State:** Occupancy and flow of each link, and rate of change of flow in the approaches. These are measured when signal is green. **Action:** duration of green for a phase, with fixed phasing and cycle length between 60s and 120s, and offsets	Yes, but by using the hierarchies, not between neighbors	25-intersection network in Paramics	Not described but variable over time (likely undersaturated)	Not described
Campoganara and Kraus	Distributed Q-learning	**State:** Number of vehicles in each approach. **Action:** allocation of right of way, 2 phases	Yes, a distributed Q-learning	Two adjacent intersections connected	Not described but fixed and undersaturated	Not described
Richter et al	Natural actor-critic with online stochastic gradient ascent	**State:** Very comprehensive state: phase, phase duration, cycle duration, duration of other phases in cycle, bit showing if there is a car waiting on each approach, saturation level (3 posssible), and neighbor information (2 bits showing where traffic is expected from). **Action:** 4 possible phases, with the restriction that all must be called at least once in the last 16 actions	Yes, 2 bits of info showing where is traffic expected from	2-intersection network and 9-intersection network, 10x10 network (not detailed results)	Not described but variable over time	Not described
Zhang and Xie	Neuro-fuzzy actor-critic RL	**State:** Queue length and signal state. Action: duration of the phase for fixed phase sequence; for variable phase sequence actions included the phase to follow	No, but recommended for multi-agent applications	4-intersection arterial in VISSIM	Variable over time based on real data. Undersaturated	Not described
Kuyer et al	Model-based RL (with Q values) - and coordination graphs	**State:** Sum of all states of blocks in the network (which represnets all vehicles in the links). **Action:** assign right of way to a specific direction.	Yes, max plus algorithm but no RL	3-intersection, 4-intersection networks, and a 15-intersection network	Not described, but experiments with different amount of "local" and "long route" percentages, to create improvements when coordination was added	Not described
Arel et al	Q-learning with function approximation. There are central and outbound agents	**State:** For each of the 8 lanes of an intersection, the state was the total delay of vehicles in a lane divided by the total delay of all vehicles in all lanes. The central agent has access to full states of all intersections. **Action:** any of 8 possible phases (an action is taken every 20 time units)	Yes, all intersections share the state with a central agent	5-intersection network with a central intersection that has the learning capabilities	Variable, including oversaturation	10000 time steps before stats were collected. In operational mode the exploration rate was 0.02

Table 2.1 – Summary past research on RL for traffic control – States and Actions

Author	Algorithm	Reward	Communication	Application	MOEs Analyzed
Thorpe	SARSA with eligibility traces	Negative values for each time step until it processed all vehicles in a time period	No	4x4 network different loads	Number of steps to process demand, average wait time per vehicle, and number of stops
Thorpe	SARSA with eligibility traces	Negative values for each time step until it processed all vehicles in a time period, positive values for every vehicle crossing the stop bar, and negative values for vehicle arriving at links with red	No	4x4 network different loads	
Appl and Brauer	Q-function approximated by fuzzy prioritized sweeping	Squared sum of average divdied by max density of links (the lower and the more homogeneous the better)	No	Single intersection	Total average density per day
Wiering	Model-based RL (with Q values)	If a car does not move, assing a value of 1, otherwise assing 0 (in sum, maximizing car movement/ or throughput).	Yes (shared knowledge or "tables" in some scenarios). Also, included a look-ahead feature	2x3 network	Throughput
Bingham	Neurofuzzy controller with RL (using GARIC, an approach based on ANN)	Delay of vehicles + V value at time t - V value at time t-1 (V depends on the approaching vehicles in links with green plus those with red	No	Single intersection	Average vehicle delay
Gieseler	Q-learning	A reward resulting from the difference in the activation times of vehicles being processes (headways) - the shorter headways the better. Also, a fraction of the rewards of adjacent intersections was added to the agent's reward	Yes, boolean variable showing if the signal was green "q" seconds earlier. Also shred information of the rewards	3x3 network	Not described
Nunes, Oliveira	Heterogeneous (some agents use Q-learning, others hill climbing, simulated annealing, or evolutionary algorithms). Then, the learning process is RL + advice from peers	Not described	Yes (advice exchange): communicate state and the action that was taken by the advisor agent, the present and past score	Single intersection - each agent controls one intersection but they are not connected	"Quality of service" as 1- sum(average time per link/average time in link of all links)
Abdulhai	Q-learning (CMAC to store Q-values)	Delay between succesive actions. Combination of delay and throughput or emissions is recommended for future research.	No, but recommended by sharing info on state and on rewards from a more global computation	Single intersection	Average delay per vehicle
Choi et al	RL agents using fuzzy sets. Three hierarchies well defined	Based on previous state as follows: (factor*(current-previous))-(current-best). Therefore it is positive if current state is greater than previous and the first parenthesis is greater than the second. A critic in the system also evaluates the performance in terms of delay	Yes, but by using the hierarchies, not between neighbors	25-intersection network in Paramics	Average delay per vehicle and time vehicles were stopped
Campoganara and Kraus	Distributed Q-learning	Not described	Yes, a distributed Q-learning	Two adjacent intersections connected	Average number of waiting vehicles
Richter et al	Natural actor-critic with online stochastic gradient ascent	Not described, but likely to be related with the number of cars in the links	Yes, 2 bits of info showing where is traffic expected from	2-intersection network and 9-intersection network, 10x10 network (not detailed results)	Normalized discounted throughput (to encourage vehicle discharge as soon as possible)
Zhang and Xie	Neuro-fuzzy actor-critic RL	Linear combination of vehicles discharged, vehicles in queue, number of new vehicles in queue, vehicles with green, and vehicles with red	No, but recommended for multi-agent applications	4-intersection arterial in VISSIM	Average delay, average stopped delay and average number of stops
Kuyer et al	Model-based RL (with Q values) - and coordination graphs	Sum of changes in network blocks: zero value if state changed, and -1 if state did not change - or vehicles did not move	Yes, max plus algorithm but no RL	3-intersection, 4-intersection networks, and a 15-intersection network	Average waiting time, ratio of stopped vehicles, and total queue length
Arel et el	Q-learning with function approximation. There are central and outbound agents	Based on the change in delay between the previous time step and the current one, divided by the max of previous or current	Yes, all intersections share the state with a central agent	5-intersection network with a central intersection that has the learning capabilities	Average delay per vehicle and percentage of time there was blocking

Table 2.2 – Summary past research on RL for traffic control – Rewards and MOEs

CHAPTER 3. METHODOLOGY

The traffic signal control problem can be defined as a system that evolves over time based on a complex stochastic process. The system behavior depends on a wide variety of combination of driver and vehicle types that produces a series of stochastic trajectories for identical initial conditions. Driver characteristics such as reaction times, acceleration and deceleration rates, desired speeds, and lane changing behavior are examples of stochastic variables that directly affect the evolution of the system state over time.

Thus, modeling the traffic state as a stochastic process, and more precisely as a stochastic process that follows the Markov property, the control of the traffic signals can be described as a Markov Decision Process (MDP) and there is potential for finding optimal or near-optimal solutions using RL strategies. In this study, two algorithms are used: Q-learning and ADP using a post decision state variable. These algorithms are very convenient to address processes with sequential decision making, do not need to compute the transition probabilities, and are well suited for high dimensional spaces (Powell, 2010).

Two separate systems were created, one for Q-learning and one for an ADP algorithm using the post-decision state variable, and the two were tested under the same conditions. Even though the formulation of the state representation and reward functions was similar for the two algorithms, the two were implemented separately.

As opposed to more traditional adaptive approaches that rely on predictions from traffic models to estimate the state of the system over time, RL agents act and then observe the performance of the actions to create knowledge, thus the process is not a predictive one, but a learning one based on the past behavior of the system. In addition, communication between agents will be allowed, such that potential blockages due to downstream congestion can be avoided, and more explicit coordination mechanisms can also be adopted.

Assuming that the system state follows the Markovian memory-less property and that the values of all future states are given (based on discounted rewards) the Bellman equation shows that the value of a given state (*s*) can be expressed based on the value of the potential states following the immediate action and the cost to get there, as follows:

$$V^{\pi}(s) = \sum_{x} \pi(s,x) \sum_{s'} P_{ss'}^{x} \left(C_{ss'}^{x} + \gamma V^{\pi}(s') \right)$$

where $V^{\pi}(s)$ is the value of state *s* following policy π (also known as the "cost-to-go"), *x* is an action drawn from a finite set of possible actions, $P_{ss'}^{x}$ is the probability of transitioning to state *s'* given that the current state is *s* and the action taken is *x*, $C_{ss'}^{x}$ is the cost of such transition, and γ is a discount factor for the value of the next state $V^{\pi}(s')$ (Sutton and Barto, 1998). Note that in the first summation *π(s,x)* is simply the probability of taking action *x* given that the current state is *s*, and that the second summation is also commonly expressed as an expectation (instead of the sum of weighted values) for taking action *x*.

Thus, based on this representation of the state value, it is possible to formulate an optimization problem in order to find optimal state values, which in turn represents the problem of finding an optimal policy ($V^{*}(s)$):

$V^{*}(s) = max_{\pi} V^{\pi}(s), for\ all\ s \in S$, or

$$V^{*}(s) = \max_{x} \sum_{s'} P_{ss'}^{x} \left(C_{ss'}^{x} + \gamma V^{*}(s') \right)$$

However, since the true discounted values of the states are not known (otherwise finding optimal policies would not be a problem) some algorithms have been used to solve this problem both in an exact and an approximate fashion. The most well known exact strategy is the traditional dynamic programming (DP) approach originally proposed by Richard Bellman, and approximate methods emerged well after (in 1980s), including temporal difference methods (TD).

Traditional DP is a very powerful tool that can be used to solve the Bellman equation and guarantees the optimality of the solution. However, the number of required

computations using a standard DP algorithm grows exponentially with a linear increase in the state space, the output space, or the action space, deeming it intractable for real-sized problems. This is known as the curse of dimensionality of DP, and can be described as the need to perform nested loops over the state and action space as the algorithm finds the state values in a backward recursion.

To illustrate the curse of dimensionality in a centralized signal system, consider the task of finding the optimal signal timings for a period of 15 minutes (assuming the control is evaluated every 10 seconds, which is a very coarse approximation) in a network with only 10 intersections, and each of them can display up to four different phases (through and left-turn movements for E-W and N-S directions). Also assume that the demand for each movement can be categorized in 20 levels, thus if the capacity of the link is 60 vehicles some loss of resolution is allowed. This leaves us with a combination of 20^4 states per intersection (assuming 4 links, thus a combination of 20x20x20x20) and 20^{40} (20^4 combined for the 10 intersections, thus 20^{4*10}) for the whole system at a single point in time. If the signals are re-evaluated every 10 seconds, a total of 90 decisions points are required. This makes any backward recursion intractable, as looping through the state space at every decision point is clearly unfeasible in practice.

Moreover, DP algorithms need a complete model of the systems dynamics (or transition function) in order to perform a backward recursion and estimate the optimal state values. However, the precision of traffic model predictions decrease as the prediction horizon increases, indicating that if DP is used the solutions will be built backwards starting from the least accurate end of the horizon.

On the other hand, compared to other methods to solve RL problems (i.e. dynamic programming and Monte-Carlo methods), TD learning methods are well suited for real-time traffic control applications since they combine the following features: a) Learning can be performed without knowing the dynamics of the environment, b) estimates are based on previous estimates (bootstrapping) so there is a solution for every state at every point in time (i.e. any-time algorithm), and c) they use forward-moving algorithms than can make use of real-time inputs as the system evolves. These are precisely some of the

main reasons why algorithms using TD methods are practical for the problem of managing traffic signals in a traffic network.

Standard TD algorithms are designed to learn optimal policies for a single agent, given their perceived state of the system. However, since the perceived states are typically confined to the immediate surroundings of the agent (e.g. vehicles in the approaches of the agent's intersection), changes in the dynamics of neighboring agents could make the learned policies no longer optimal. These characteristics emphasize the importance of a precise state representation to capture the dynamics of the environment, allowing for adequate learning and communication between agents in order to promote signal coordination.

Depending on the coverage of a single agent and its perception limits, several RL traffic control structures can be defined including three obvious cases: a) a single agent that directly controls all intersections of a traffic network (completed centralized); b) few agents, each controlling a group of intersections (partially decentralized); and c) one agent per intersection (completely decentralized). Options *a* and *b* may suffer from prohibitive number of states per agent, in addition to the increased vulnerability of the system in case of an agent failure. On the other hand, option *c* seems more appropriate, it may have better scalability properties for large systems, is less vulnerable, and (not surprisingly) it has actually been pursued by most researchers using RL techniques for traffic control.

Out of a handful of TD algorithms, Q-learning and ADP with the post-decision state variable will be used to find near optimal signal timings in traffic networks. The selected algorithms move forward in time to improve the updates of the values of being in each state (or "cost-to-go"), which then are used as a decision-making tool.

A more detailed description of Q-learning and ADP with the post-decision state variable are provided next.

3.1 The Q-learning Algorithm

As described above, the RL problem can be thought as the problem of finding the policy that guarantees maximum expected rewards:

$$V^*(s) = max_\pi V^\pi(s), for\ all\ s \in S$$

This maximization problem can also be described in terms of the value of state-action pairs (called Q-values), and therefore the goal will be to find a policy with action-value functions ($Q^\pi(s,a)$) leading to maximum expected total rewards:

$$Q^*(s,a) = max_\pi Q^\pi(s,a)$$

The advantages of having values of state-action pairs, as opposed of only states, are mostly observed in systems where the dynamics are not completely known (the algorithm is model-free) or the random information received over time is not precisely determined in advance. The reason for such advantage is that there is no need to estimate the full expectation of the transition function to perform an update of the Q estimates (as opposed to the standard Bellman equation):

$$\hat{q}(s,x) = c^x_{ss'} + \gamma \max_{x'} Q(s',x')$$

as opposed to

$$Q(s,x) = C^x_{ss'} + \gamma \sum_{s'} P^x_{ss'} \max_{x'} Q(s',x')$$

Since the learning process is done gradually and based on experiencing sampled information from the system, the estimates can be updated using the following standard rule:

$$Q(s,x) = (1-\alpha)Q(s,x) + \alpha\hat{q}$$

Where α is the learning rate.

The general algorithm for Q-learning can be formulated as shown in Figure 3.1 below.

1) Initialization:

Q⁰(s,x)=y, y is and arbitrary value

2) While *n* episodes not finished:

Find initial state (s_0^n)

While episode not finished (t<T):

Select action x based on a given policy

Execute x, observe $c_{ss'}^x$, s'

Update: $Q_{t-1}^n(s,x) = (1-\alpha)Q_{t-1}^{n-1}(s,x) + \alpha(c_{ss'}^x + \gamma \max_{x'} Q_t^{n-1}(s',x'))$

Advance to next step: $s = s'$

Figure 3.1. Pseudo-code for the Q-learning algorithm

Q-learning has shown good performance for a variety of practical problems under stationary conditions, even though the convergence of Q-values has only been proven if the states are visited an infinite number of times (Watkins, 1989, 1992). This is because practical decision making does not require full convergence of Q-values as long as they are "sufficiently" different for the agent to commit to the best choice. Unfortunately, precise boundaries of the Q-learning algorithm for decision-making purposes only are not well defined and require further research.

3.2 The ADP Algorithm with Post-decision State Variable

Unlike standard DP, which finds the best policy from exact values of the states, ADP uses approximate state values that are continuously being updated. Estimates of state values are available at any point in time (thus, the algorithm is suitable for real-time control), and bootstrapping is used for closing the gap between approximate estimates and the true value of a state (similar to Q-learning). Also, since ADP does not require a model of the dynamics of the system over time, the system moves step by step following a transition function (that does not need to be known) provided by a simulation environment or by incoming real-world data.

There are a series of variants to the basic ADP algorithm, but for this research it was decided to adopt an ADP algorithm that uses the "post-decision" state variable; more

precisely, the formulation described by Powell (2007). This algorithm provides a series of computational advantages, as it is explained below.

The post-decision state variable formulation uses the concept of the state of the system immediately after an action is taken. This will be described based on the expression that represents the transition function of our problem:

$$S_{t+1} = S^M(S_t, x_t, W_{t+1})$$

Where the state changes from S_t to S_{t+1} in a transition that starts at time t and ends at $t+1$. W_{t+1} represents the exogenous (or random) information that influences the transition from state S_t to S_{t+1}, after executing action x_t. Specifically for our system, the exogenous information is the combination of different driver and vehicle characteristics that ultimately translates in the (stochastic) behavior of vehicles in the traffic stream.

Note that the transition shown above can be also described by the following sequential steps:

1) The system has just arrived at time t and the state (S_t) has been updated based on the transition from the last time step:

$$S_t = S^M(S_{t-1}, x_{t-1}, W_t)$$

2) Also at time t, the state of the system (S_t) is modified immediately after the action x_t is taken (S_t^x), but no exogenous information from time t to $t+1$ has been received (in other words, the signal has just changed but vehicles have not reacted to it):

$$S_t^x = S^{M,x}(S_t, x_t)$$

3) At time t+1, the exogenous information (W_{t+1}) has been received and the transition from S_t^x to S_{t+1} has been completed (this is, after the vehicles have reacted to the signal):

$$S_{t+1} = S^{M,W}(S_t^x, W_{t+1})$$

Similarly, the process to update the value of a state from one time step to the next can be decomposed as follows:

1) The value of state S_{t-1} at time *t-1* after committing to action *x*, S_{t-1}^x, can be expressed as a function of the expected value of the next state $V_t(S_t)$, following the Markov property:

$$V_{t-1}^x(S_{t-1}^x) = E\{V_t(S_t) \mid S_{t-1}^x\}$$

2) In addition, the value of the next state (at time *t*) can be expressed based on the maximum value of the state after taking the optimal action X_t (this is, $V_t^x(S_t^x)$) and the cost to get there C_t:

$$V_t(S_t) = \max_{x^t}\left(C_t(S_t, X_t) + \gamma V_t^x(S_t^x)\right)$$

3) Analogous to the expression in step 1, the sequence repeats for the value of state S_t, but at time *t* and after committing to a new action *x*:

$$V_t^x(S_t^x) = E\{V_{t+1}(S_{t+1}) \mid S_t^x\}$$

As explained in Powell (2007), the standard optimality equation could be obtained by combining the equations in steps 2 and 3. However, if the first two equations (steps 1 and 2) are combined instead, a new expression using the "post-decision" state variable is obtained as follows:

$$V_{t-1}^x(S_{t-1}^x) = E\left\{\max_{x_t}\left(C_t(S_{t-1}^x, x_t) + \gamma V_t^x(S_t^x)\right) \middle| S_{t-1}^x\right\}$$

Note that this expression is very different from the traditional optimality equation mainly because the expectation is outside of the optimization problem.

Similar to Q-learning, this provides an important computational advantage and allows the algorithm to provide better approximate solutions as the number of iterations increases. It also allows for the use of a forward algorithm so that it is no longer needed to loop though all possible states. However, it is required to approximate the expectation of the value function. Thus, as long as the states are visited with some frequency, it is possible to have "good enough" estimates for adequate decision making support.

The value function using the post-decision variable can be updated using a similar equation as the one from the traditional update rule for temporal difference learning, as follows:

$$\bar{V}_{t-1}^{n}(S_{t-1}^{n}) = (1-\alpha_{n-1})\bar{V}_{t-1}^{n-1}(S_{t-1}^{n}) + \alpha_{n-1}\hat{v}_{t}^{n}$$

Where $\bar{V}_{t-1}^{n}(S_{t-1}^{n})$ is the approximated value of the state S_{t-1}^{n} at iteration n, and α is the step size or learning rate. The step size determines the weighted value of the current direction pointed out by $\hat{v}_t^n$ in relation to the approximation of the state value at the current iteration.

It is pointed out that since it is necessary to have a value of $\bar{V}_t^n(S_t^n)$ for each state S_t^n, the problems do not reduce their dimensionality when using ADP, but rather the number of computations needed to find an approximate solution.

The general algorithm for the ADP using the post-decision state variable is shown in Figure 3.2.

1) Initialization:

V^0=y, y is and arbitrary value

2) While *n* episodes not finished:

Find initial state (s_0^n)

While episode not finished (t<T):

Select action *x* based on a given policy

Execute *x*, observe *r*, s^x *(post-decision state variable)*

Update: $V_{t-1}^n(s_{t-1}^x) = (1-\alpha)V_{t-1}^{n-1}(s_{t-1}^x) + \alpha(\max_x(c_{s_{t-1}}^x + \gamma\bar{V}_t^{n-1}(s_t^x)))$

Observe random information (W_{t+1})

Complete transition by finding $s'(s_t^x, W_{t+1})$

Figure 3.2. Pseudo-code for the ADP algorithm using the post-decision state variable

To achieve convergence, the learning rate should decrease over time. Rules for the algorithms to converge require the same typical rules for stochastic gradient algorithms: 1) the step size should not be negative, 2) the infinite sum of step sizes must be infinite, and 3) the sum of the square of the step sizes must be finite.

3.3 An algorithm for signal coordination – The Max-Plus

A shortcoming of learning agents acting separately is that each of them will strive to take actions that maximize their local payoff without considering the global payoff of all agents together. For the problem of managing the traffic signals, agents will take actions to improve one or a combination of measures of performance such that their own set of indicators is improved over time. These measures may include throughput, delay, number of stops, or any other traffic related indicators.

Traffic networks with high demands may evolve into states that are not able to process traffic at their capacity due to oversaturation and possibly into de-facto red and gridlocks. Under these conditions, agents operating solely on the basis of their approaching links may take decisions that could degrade the performance of adjacent intersections and ultimately its own. For example, an intersection at the edge of a network may allow vehicles to enter at a rate that is higher than the rate that can be processed

downstream due to high conflicting volumes. This situation will eventually create queue overflows inside the network and a gridlock, which may translate into a great decrease in the network throughput.

The scenario described above can be easily encountered when demands are high, and particularly in situations where the conflicting volumes are also high. Therefore, if the traffic system is controlled by agents operating individual intersections, they should be able to communicate with each other at least to a certain degree.

A series of communication capabilities can be thought to be important. A first level of communication could be the transmission of information regarding the current state of neighboring intersections. In this way, for example, an agent could prevent queue overflows in downstream links or even create green waves to reduce the number of stops of oncoming platoons. As it can be imagined, this simple mechanism can create emergent coordination, as it was explored by Medina et al. (2010) for the case of an arterial.

A second level could be related to not only the transmission of the neighboring states, but also of knowledge stored in the form of learned policies (e.g. Q or V values). The communication in this case can be perceived as an advice exchange from agents with similar characteristics and geometry.

In addition, a more explicit mechanism can be implemented such that the exchange is not limited to information-passing only, but is extended to the actual decision-making process. For example, a coordination strategy can be devised for agents to take decisions as a team or a coalition considering network-wide implications. A series of algorithms have been proposed to obtain “negotiated” actions using either hierarchy of agents (given assigned priorities) or through message passing among agents at the same level of influence, with “variable elimination” being one of the most widely known because it is guaranteed to converge, however it is not an any-time algorithm and solutions are not available until all iterations have been completed.

On the other hand, in this category of explicit mechanisms for coordination the max-plus algorithm (Vlassis et al., 2004; Kok and Vlassis, 2005 and 2006) emerges as a

viable option for controlling the traffic signals in a traffic network. The max-plus algorithm uses a message-passing strategy that is based on the decomposition of the relations in a coordination graph as the sum of local terms between two nodes at the time. This allows the interchange of messages between neighboring intersections, such that in a series of iterations the agents will reach a final decision based on their own local payoff function as well as the global payoff of the network.

Thus, the max-plus is an algorithm that propagates the combination of local and global payoffs among the agents that are interconnected in a coordination graph. Locally optimized messages $U_{ij}(a_j)$ are sent by agents *i* to neighbor *j* over the edges that connect them and with respect to the action executed by agent *j* (a_j). For tree structures, the algorithm converges to a fixed point after a finite number of iterations (Pearl, 1988; Wainwright et al., 2004). However, proof of convergence is not available for graphs with cycles, and there is no guarantee on the quality of the solution of max-plus in these cases. Nonetheless, as pointed out by Kok and Vlassis (2006), the algorithm has been successfully applied in practice in graphs with cycles (Murphy et al., 1999; Crick and Pfeffer, 2003; Yedidia et al., 2003).

Kok and Vlassis (2006), describe the algorithm and some considerations when it is applied to graphs with cycles. For the traffic signal problem, and in particular for grid scenarios, the intersections are interrelated by connections in all their approaches and create a series of cycles between neighboring intersections. The algorithm and the considerations for graphs with cycles are described next.

Let's suppose that the traffic network is a graph with |V| vertices (or intersections) and |E| edges (or links). To find the optimal action in the network (a^*), agent *i* repeatedly sends the following message u_{ij} to its neighbors *j*:

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

Where Γ(*i*)*j* are all neighbors of *i* except *j*, and c_{ij} is a normalization value. Message u_{ij}, as explained by Kok and Vlassis (2006), is an approximation of the maximum payoff agent *i* can achieve with every action of *j*, and it is calculated as the sum of the payoff functions f_i, f_{ij}, and all other incoming messages to agent *i*, except that from agent *j*. Messages u_{ij} are exchanged until they converge to a fixed point or until the agents are told to stop the exchange due to an external signal, for example after the time available to make a decision is over. It is noted that the messages only depend on the incoming messages of an agent's neighbors based on their current actions, thus there is no need to have these messages optimized, nor evaluated over all possible actions.

On the other hand, the normalization value c_{ij} is very useful specially for graphs with cycles since the value of an outgoing message u_{ij} eventually becomes part of the incoming message for agent *i*. Thus, in order to prevent messages to grow extremely large, it is proposed to subtract the average of all values in u_{ik} using:

$$c_{ij} = \frac{1}{|A_k|}\sum_k u_{ik}(a_k)$$

For this study, given that the agents are implemented in a microscopic traffic simulator where the states are updates in a synchronous fashion, the centralized version of the max-plus algorithm was implemented. This version has been taken from Kok and Vlassis (2006) and it is shown in Figure 3.3.

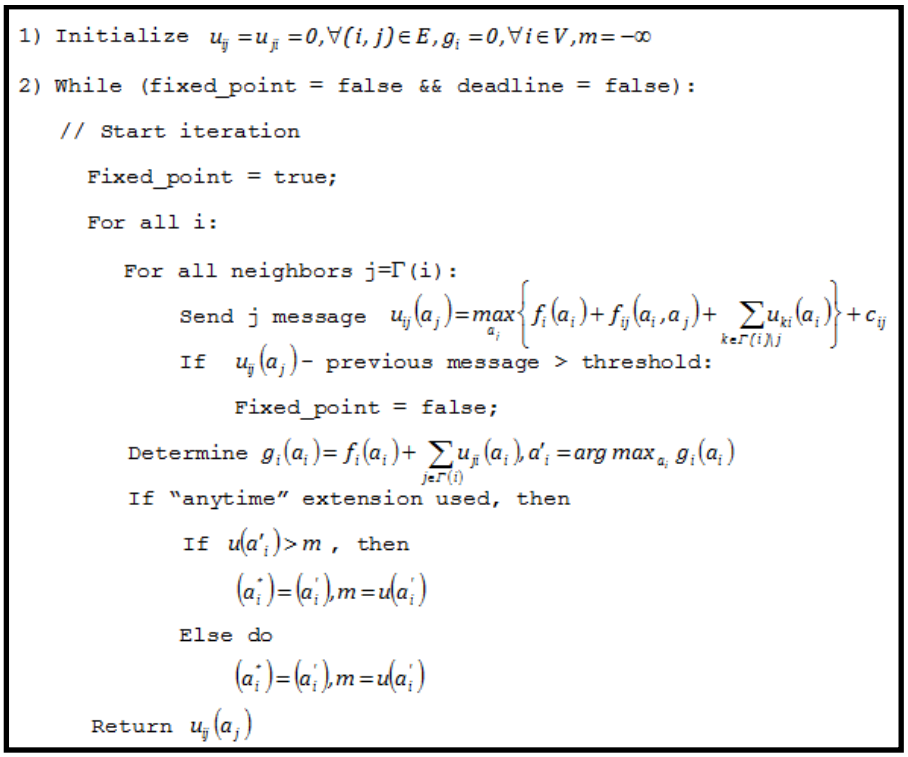


Figure 3.3. Pseudo-code for the centralized max-plus algorithm from Kok and Vlassis (2006)

3.4 Implementation of the algorithms for real-time traffic control

The proposed traffic control is completely decentralized and relies on independent agents with communication capabilities. The general structure of an agent and its interaction with the traffic environment is represented schematically in Figure 3.4. As it is typical of an agent-based application, the only direct input from the environment to the agent is in the form a "perceived" state, which in this particular case comes from static traffic sensors and the state of the traffic signals, in addition to an indirect input of the environment through communication with other agents. Conversely, the only mechanism for the agent to impact the traffic environment is through actions that modify the status of the traffic signals and the actions of other agents.

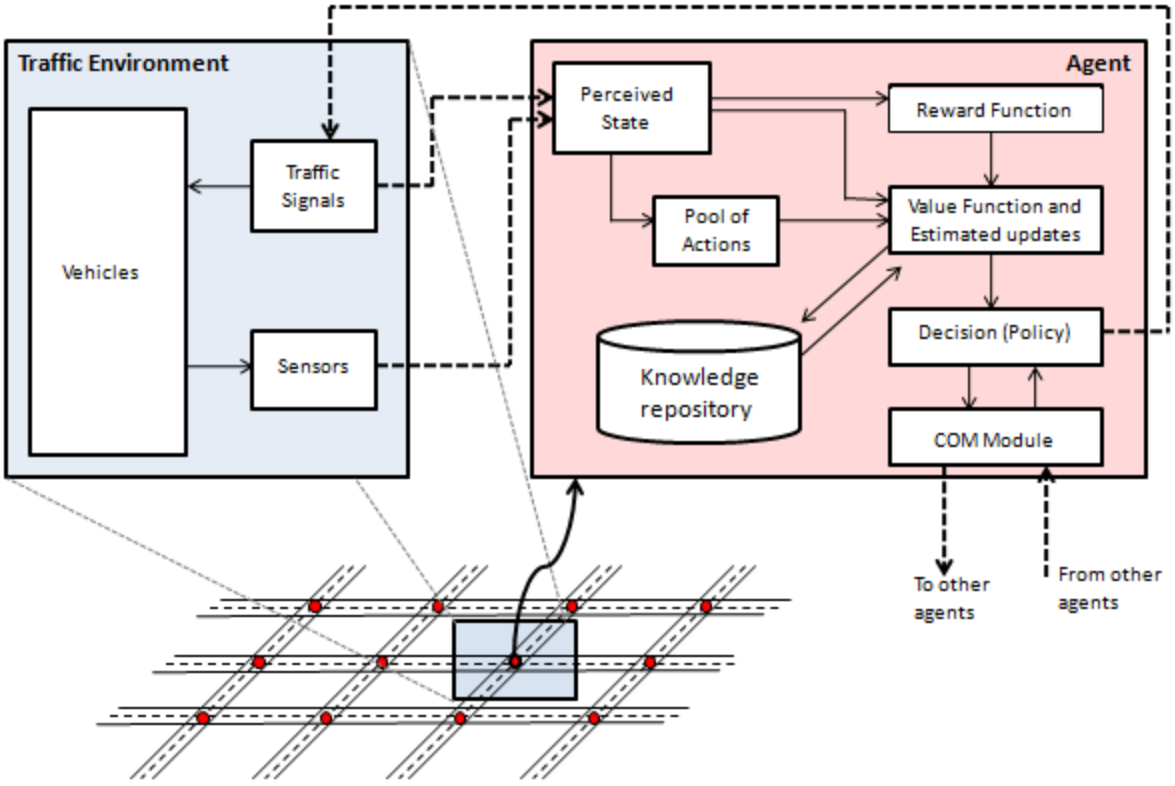


Figure 3.4. Schematic Representation of Agent and Environment Structures

Inside the agent structure, there is close interaction among all elements. The structure is standard of an agent using a RL algorithm, with exception of the COM module (for achieving explicit coordination and signal progression), and can be described in general using Figure 3.4 as follows.

Information from the environment is received by the agent and recognized as the current state of system. The state is used in these internal tasks: 1) estimation of the reward of the previous action, 2) determination of potential actions from a pool of all possible actions, 3) estimation of the value of the previous state (or state-action pair), and 4) communication with other agents. The estimation of the reward requires a comparison between previous state(s) and the current one (to determine the goodness of the action), and the execution of a model to determine changes in the desired measures of performance (e.g. delay or amount of carbon emissions). The value of being in a given

state (for Q-learning the value is also associated to the action) uses the reward estimation and previous knowledge on the value of the state. A value function provides estimates of the “true” or discounted value of a state (or a state-action pair), which is also known as the “cost-to-go”.

It is noted that the agent determines the best action and the learning is based on the optimal choice, but the algorithm does not force the agent to commit to such action. This is the reason why Q-learning and the selected ADP algorithm are called “off-policy” - since the learning process is optimal even though the policy may continuously change, for example by following an exploration strategy instead of always using an *e*-greedy criterion.

Finally, the agent commits to a decision based on the learned policy. This step may be influenced by the results of the value function and the information exchange with other agents, provided that the action selected belongs to the set of valid actions; The selected actions are sent to the traffic signals for their execution, which in turn will affect the vehicular traffic (and the state of the system) and the cycle starts again by the agent observing a new state.

The RL agents were coded and tested in a C++ computer environment and coupled with the traffic simulator VISSIM, produced by PTV AG. This simulator is able to provide the desired undersaturated and oversaturated traffic conditions and allows for the state of the traffic signals to be manipulated in running time based on user-defined rules. This is done through a communications interface in VISSIM, and the access to the simulation is accomplished using a dynamic linked library (DLL), generated by the custom C++ code.

As described earlier, each intersection is operated by a single agent. Therefore, each agent has its own separate set of state values and keeps track of its own gained knowledge independently. Each agent in VISSIM sequentially calls the DLL every simulation second, thus all variables accessible to the user can be tracked with the same frequency. The current implementation updates the agents every two seconds, given that this is the typical time needed to process a single vehicle through an intersection at

saturation flow rate. In addition, a 2-second window for evaluating the next signal response is expected to provide very accurate results, also leaving more time available to other functions that may be needed, such as communication between agents and conflict resolution for group formation.

Currently, default driver behavior parameters from VISSIM have been used in the simulations, since at this point the objective was to determine the feasibility of the methodology, not the precise representation of a particular network in the real world. Nonetheless, some work has been done to calibrate VISSIM parameters and compare results obtained by the agents with other commercially available traffic signal optimizers such as TRANSYT7F implemented in a different simulation package (CORSIM).

The information that agents receive about the state of the system is collected via vehicle detectors placed along the roadway. This allows for calculations of the number of vehicles in the links, queues, density, and speeds in all approaches, which the agents can use to make the control decisions (in addition to information received from other agents).

Recall that the agents can operate with complete flexibility in terms of timing parameters, thus the operation of the signals is not restricted by pre-specified cycle length, splits, or offsets. Furthermore, restrictions such as maximum green times or phase sequence, are not an issue in this implementation. Nonetheless, a minimum green time of 8 seconds was deemed reasonable and was imposed in all experiments in this study.

3.5 Experimental Setup

Different experimental scenarios were defined and created in VISSIM to determine the performance of the algorithms. The analysis of the algorithms is focused on both undersaturated and oversaturated conditions and a series of measures of performance were used to evaluate the effects of using variations of Q-learning and ADP parameters.

Experiments were defined such that the level of complexity was increased, starting from a single intersection, followed by two scenarios with arterials, and finally evaluating a medium-sized network with 20 intersections, as described below. Figures for

all the scenarios are shown in the next chapter, immediately before the analysis of their results.

- Single intersection, oversaturation: this scenario required the algorithms to control a single intersection with four phases (exclusive through and left-turn movements), where the demands for all approaches and phases is very high. The intersection was assumed to be isolated and had long entry links (2000 ft) and long left-turn lanes (1000 ft), all of which had a single lane. The traffic signal controlling the intersection could display up to four phases, two for the through movements and two for the left movements. The phase sequence did not have any restrictions. Traffic demands ensured oversaturation, with 1000 vphpl for each of the four entry links and 20% of such demand turning left. The schematic representation of the single intersection is shown in the following Chapter, Figure 4.1.

- Four-intersection arterial, undersaturation: this second scenario was designed to test the potential coordination of four closely spaced intersections along a corridor. Two of the intersections did not have conflicting demand, therefore the agents should learn not to provide green time to the unused approaches. In addition, the other two intersections had demand in all approaches and were next to each other, thus it was possible to coordinate their actions to improve the performance of the arterial. The arterial had two lanes per direction and short left-turn pockets of approximately 140 ft. The distance between intersections 1 and 2, and between 3 and 4 was close to 400 ft, and there was a link 790 ft long between intersections 2 and 3. The entry links on the arterial were about 500 ft long and the approaches in the opposite direction had 3 or 1 lane per direction. Entry volumes were 1000 vphpl in the north and south bounds at both ends of the arterial, and one third (333 vphpl) for the conflicting movements in intersections 1 and 2.

- Two parallel arterials: this scenario contains a total of 10 intersections where two two-way streets run parallel to each other and intersect streets with high conflicting demands. This is a step up in the complexity level for finding traffic signal timings compared to the previous scenario, as there is interaction between intersections in

both directions of traffic. Both undersaturated and oversaturated conditions were analyzed in two separate cases. Demands for the oversaturated case were at similar levels than in previous scenarios, with 1000 vphpl, and for the undersaturated case they were 1000 vphpl in the direction with high demand and about one third of this amount (333 vphpl) for the opposing direction.

- Medium-sized, realistic network in undersaturation and oversaturation: a network of 20 intersections was created based on a section of downtown Springfield, Il. This network is an expansion of the previous scenarios, and the intersections are distributed in a 4x5 grid-like configuration. This scenario is highly complex, as there are combinations of one-way and two-way streets, as well as different number of lanes. This scenario can be regarded as a realistic one, where the potential for ADP and Q-learning can be observed in challenging conditions in terms of traffic control. Both undersaturated and oversaturated conditions were analyzed.

CHAPTER 4. ANALYSIS OF RESULTS

As described above, a series of experiments were designed to determine the performance of the reinforcement learning algorithms in scenarios with increased complexity, beginning with a single isolated intersection, then for two different arterials, and lastly using a mid-sized realistic network. This section describes the results and presents an analysis of the algorithms in these scenarios in terms of a series of a performance measures or indicators including: vehicle throughput, delay, number of stops, signal timings, queues, and average discharge headways.

This set of indicators provides a clear understanding of the behavior of traffic when the intersections were controlled by the algorithms. The case of the single intersection is presented next, followed by the remaining cases in order of complexity.

4.1 Single Intersection - Oversaturated Conditions

As described above, a single intersection with one lane per approach and exclusive left-turn lanes was created for the first scenario. A sample image of the single isolated intersection in VISSIM is shown in Figure 4.1.

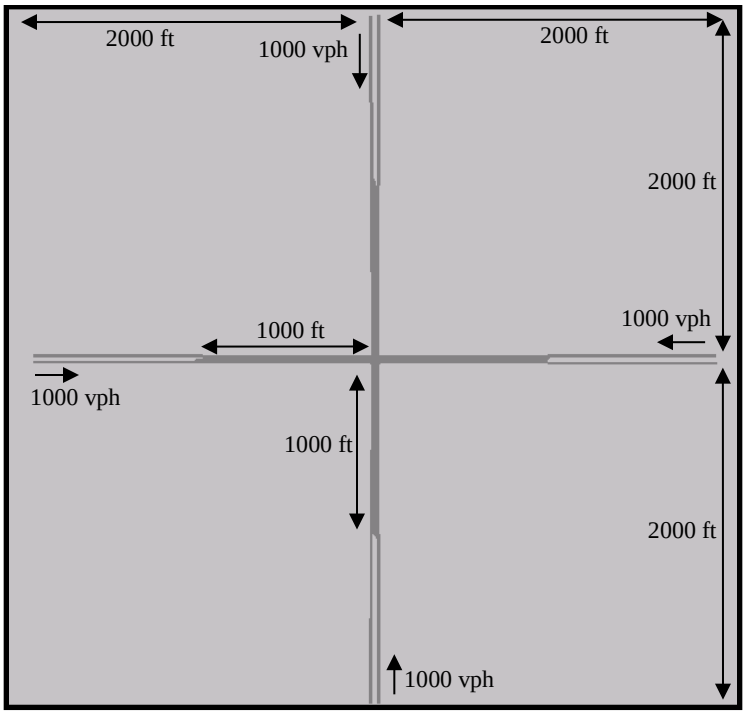


Figure 4.1. Schematic representation of single isolated intersection

The agents were trained during 160 replications of 15 minutes each, where they accumulated experience and improved their performance based on the feedback received through the reward function. The number of replications was chosen after observing the learning curve of the agents, peaking near the 100th replication, so that the performance measures were obtained after the training was in its final stages. Results from the last 20 replications were used to estimate the performance of the already-trained agents.

A total of four variations of the ADP algorithm and four more of the Q-learning algorithms were implemented by incorporating different state and reward functions. Results are presented for the ADP implementation first, followed by those using Q-learning.

4.1.1 ADP implementations

Four variations were tested in this scenario to explore different state and reward representations, and their potential effects on the intersection performance. The following implementations were evaluated:

- ADP 1: The state was represented by a five-dimensional vector with one dimension for the demand of each phase and an additional dimension for the status of the current phase. The reward for displaying a given phase was also very simple and calculated as the total demand present in the approach served by this phase. A penalty for changing phases was imposed to account for the lost time in the yellow-red transitions and it was a value proportional to the demand being served by the new phase.

- ADP 2: This application used a similar state and reward representation to that in ADP 1, but included an additional component in the state that indicated the duration of the current phase being displayed. The rationale behind this additional information was to serve as a proxy for the delay of vehicles in the phases not being served. The reward structure used in ADP 1 was maintained unchanged.

- ADP 3: Instead of using the phase duration as a proxy for the delay of competing demands, this implementation used an estimation of the time that vehicles have spent in the link. This value was then combined with the actual number of vehicles to determine the state of each of the demands in the four phases. The time vehicles have been in the link was accumulated using a dynamic table that kept track of vehicles as they entered and left the link, assuming no lane changes. This information can be easily found in the field with the use of entry and exit detectors. The reward structure remained unchanged, thus the effects in the performance will reflect only those of the added information. For this implementation, phase duration was not included as a dimension in the state space.

- ADP 4: This implementation is similar to that used in ADP 3, with the exception that the phase duration was added to the state representation. The reward structure was the same as the one used in the implementations above.

4.1.2 Performance

In oversaturated conditions it is common practice to maximize the number of vehicles processed by an intersection, or vehicle throughput. For the case of a single

intersection, this may be the case because upon demands that exceed capacity, it is often desired to meet as much of such demand so that the remaining number of vehicles is as low as possible. The learning curve for the agents running the four ADP implementations is shown in Figure 4.2., where it is observed how the performance of the signal was improved over time as the agents continued accumulating experience. For the two algorithms that had the best performance (ADP 1 and 3), the throughput reached about 700 vehicles in 15 minutes for the four phases combined. This translates to about 1400 vphpl of vehicles processed by a single approach. Note that along with the actual throughput for each replication, a 10-point moving average is also displayed in Figure 4.2. for each implementation.

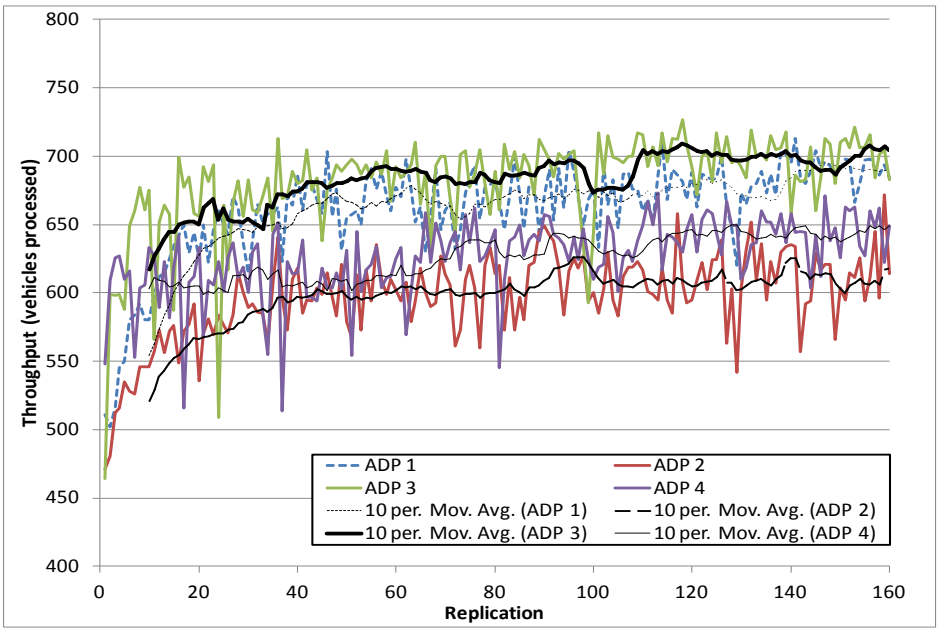


Figure 4.2. Learning curve for throughput of ADP algorithms in a single intersection

Additional analysis to determine how efficiently was the green time utilized in each phase was conducted by taking into account the signal timings. The total green time of the last 20 replications was used for this analysis in order to take into account the internal variation of the simulation software and the data when the agents had the most accumulated training time.

The average duration of each phase and their throughput for the last 20 replications is shown in Table 4.1. This allowed an estimation of the average discharge headways for each phase, which can be easily translated into green time utilization. It is observed that the lowest discharge headways were obtained using ADP 3, which makes use of the time vehicles have spent in the link as part of the state and did not include the phase duration in the state space. It is also noted that the total throughput found with ADP 3 was also the highest, confirming that this implementation had a favorable performance compared to the others, as it can also be observed in Figure 4.2. above.

Implementation	Indicator	Phase				
		Green EW Left	Green EW Thru	Green NS Left	Green NS Thru	Total Througput
ADP 1	Ave green time (s)	8.23	10.07	8.37	9.8	33451
	Total phase frequency	402	1206	429	1220	
	Total green time (s)	3308	12144	3591	11956	
	Throughput (veh)	3284	13476	3410	13281	
	Ave. discharge headway (s)	**2.01**	**1.80**	**2.11**	**1.80**	
ADP 2	Ave green time (s)	8.19	9.36	8.1	9.24	30037
	Total phase frequency	294	1168	759	1181	
	Total green time (s)	2408	10932	6148	10912	
	Throughput (veh)	2632	12130	3089	12186	
	Ave. discharge headway (s)	**1.83**	**1.80**	**3.98**	**1.79**	
ADP 3	Ave green time (s)	8.18	10.32	8.31	10.08	34174
	Total phase frequency	385	1216	385	1216	
	Total green time (s)	3149	12549	3199	12257	
	Throughput (veh)	3306	13834	3352	13682	
	Ave. discharge headway (s)	**1.91**	**1.81**	**1.91**	**1.79**	
ADP 4	Ave green time (s)	8.1	9.19	8.01	9.14	31449
	Total phase frequency	264	1250	632	1269	
	Total green time (s)	2138	11488	5062	11599	
	Throughput (veh)	2378	12794	3313	12964	
	Ave. discharge headway (s)	**1.80**	**1.80**	**3.06**	**1.79**	

Table 4.1. Signal timings and average discharge headway for ADP in a single intersection

Even though the number of vehicles processed and efficiency in the utilization of green time are important indicators of the signal performance, other indicators such as queue lengths and quality of service for all users should also be considered. For example, it would be useful knowing how fair the service is for a driver turning left compared to a

driver continuing straight through the intersection. In this regard, from Table 4.1., it is observed that the frequency with which the left-turn and the through phases were displayed was very different for all implementations, with through phases being around 3 or 4 times more frequent and with higher average duration. Recall that the demands for the left-turn phases were 20% of the total incoming traffic, thus the actual allocation of green time actually reflected the demand distribution.

Figure 4.3. shows the average vehicle delays for the four ADP implementations. Moving averages for each of the implementations help the reader observe trends for the four cases. It is noticed that the lowest average delays were obtained using ADP 1 (which had the second highest throughput), followed by those using ADP 3 which had the highest throughput. On the other hand, similar to the results from Figure 4.2. (throughput), the performance of ADP 2 and ADP 4 was (which included the phase duration) was not on par to the other two cases.

In addition, in order to determine the fairness and quality of service for left and through movements, a detailed analysis was performed on the delay of vehicles for each phase. The average and variance of the last 20 replications for left-turning and through drivers is shown in Table 4.2. for the four ADP implementations. Table 4.2. also shows the relative delay of left-turners compared to those continuing through the intersection. This can also be seen as a measure of fairness for all users.

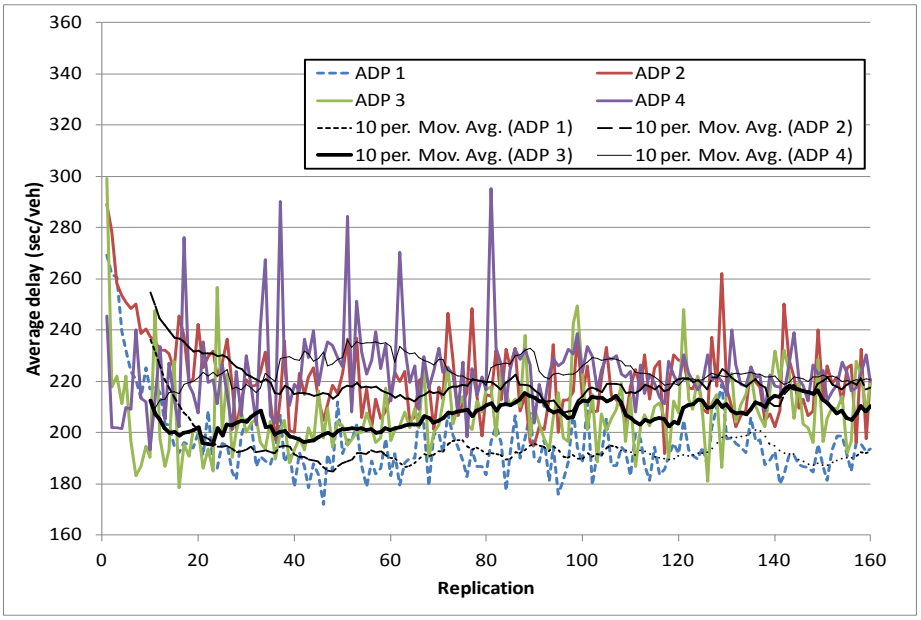


Figure 4.3. Learning curve for average delay of ADP algorithms in a single intersection

	Left-turn phases		Through phases		Ratio
Implementation	Mean (s)	Variance (s)	Mean (s)	Variance (s)	Left/Through
ADP 1	226.3	531.6	242.5	29.6	0.93
ADP 2	251.9	17758.6	283.8	4.1	0.89
ADP 3	373.5	487.3	208.5	34.8	1.79
ADP 4	395.6	89499.7	245.1	10.8	1.61

Table 4.2. Delay per phase for ADP implementations in a single intersection

The lowest delay per phase was obtained with ADP 3 for the through movements, but the most balanced service was provided using ADP 1, where the mean waiting time for both left and through movements was practically the same. Other implementations (ADP 2 and ADP 4) were highly unstable and provided longer delays for both left and through movements, and significantly higher variances for the left-turn phases.

At this point a tradeoff is observed between providing more balanced service (ADP 1) and favoring the phases with higher demands (ADP 3) but achieving the highest throughput. It is also noted that even though the differences in the average signal timings between ADP 1 and ADP 3 were very small, this resulted in significant changes in the ratio of delay between drivers in left and through phases and the throughputs.

Lastly, the average speed of all vehicles in the network is shown for the four ADP implementations. ADP 1 had the highest average speeds, which combined with the lowers average delays and the second highest throughput, provides a favorable performance along with ADP 3 which has the highest throughput. Similar to previous Figures, Figure 4.4. shows a 10-point average speed shows the learning curve as the agents gain and accumulate experience.

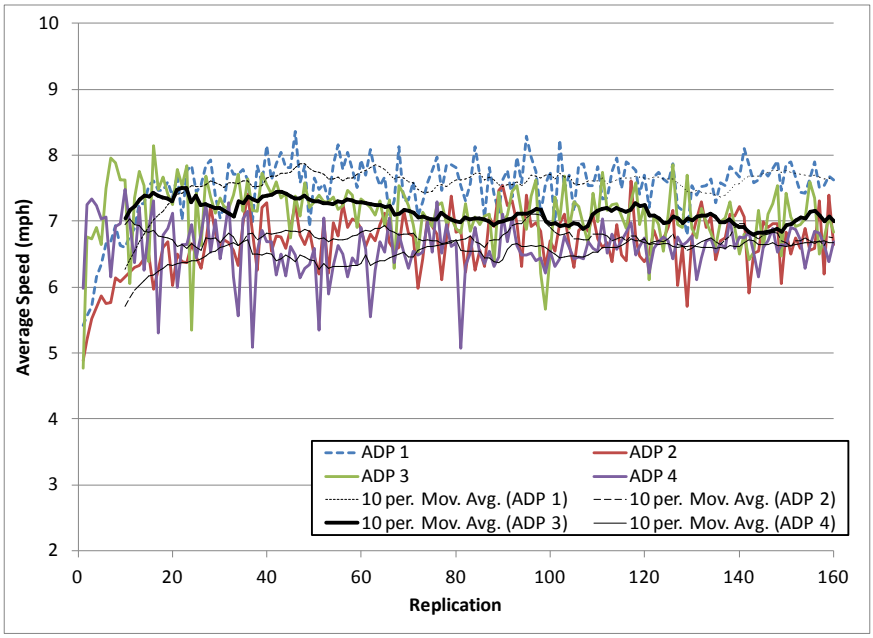


Figure 4.4. Learning curve for average speed of ADP algorithms in a single intersection

4.1.3 Q-learning implementations

A series of signal controllers were created for Q-learning algorithms, following similar implementations to those used for ADP. Thus, there were four analogous cases with Q-learning that use the same state and reward definitions as explained for ADP 1 through 4. The reader is directed to the ADP definitions in the previous subsection for details. The analysis of the performance of these implementations is described below.

4.1.4 Performance

The first indicator to determine the performance of the algorithms was the intersection throughput. The learning curve for the Q-learning implementations is shown

in Figure 4.5., where there was a distinctive improvement using Q1 and Q3, compared to those that had the phase duration as part of the state representation (Q2 and Q4). This trend is similar to that observed for ADP implementations.

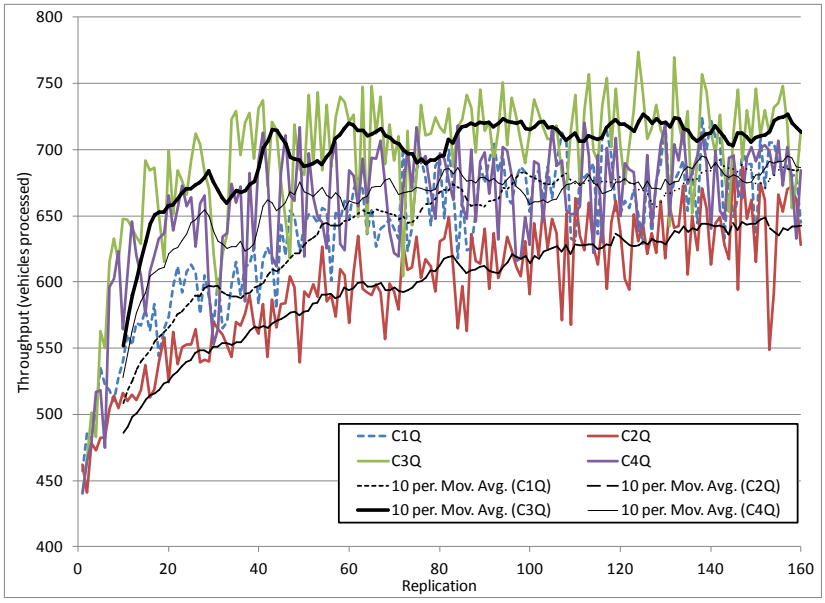


Figure 4.5. Learning curve for throughput of Q-learning algorithms in a single intersection

A direct comparison between ADP and Q-learning is also possible given that the algorithms make use of the same information from the simulation and share the source code for data collection and processing. In addition, the same random seeds were used for the two algorithms, allowing for a paired comparison. Figure 4.6. shows the two most favorable implementations for both ADP and Q-learning are implementations 1 and 3. It is easily observable that performance of the two algorithms is comparable at the end of the 160 training runs, especially for Q3 and ADP 3, reaching the highest throughput levels for the two series of implementations.

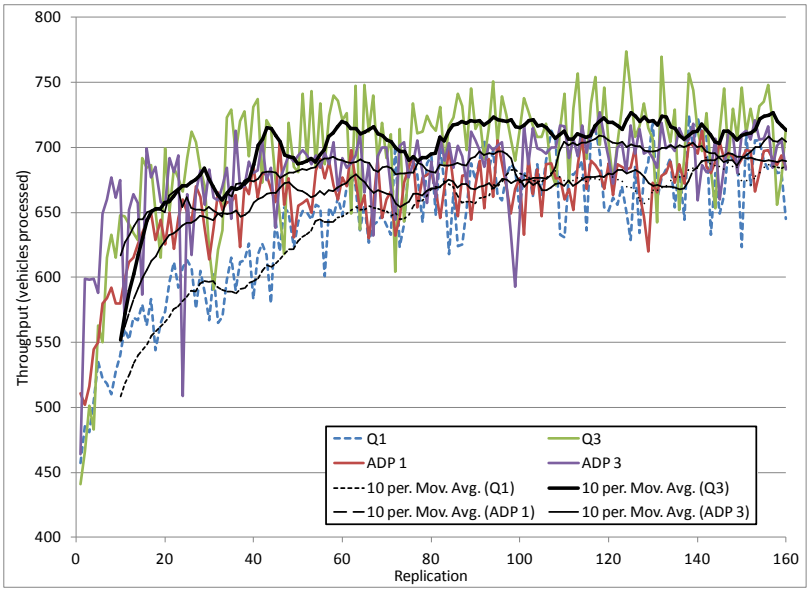


Figure 4.6. Learning curve for throughput of best Q-learning and ADP in a single intersection

The signal timings and the throughput per phase were also examined for the Q-learning implementations. From this, the average discharge headway was obtained and used as a measure of the efficiency green time utilization. Results of this analysis are shown in Table 4.3. below.

Implementation	Indicator	Phase				
		Green EW Left	Green EW Thru	Green NS Left	Green NS Thru	Total Througput
Q 1	Ave green time (s)	8	13.43	8.14	13.49	33198
	Total phase frequency	488	889	555	905	
	Total green time (s)	3904	11939	4518	12208	
	Throughput (veh)	3267	13058	3443	13430	
	Ave. discharge headway (s)	**2.39**	**1.83**	**2.62**	**1.82**	
Q 2	Ave green time (s)	8.01	10.88	8.09	10.93	31225
	Total phase frequency	505	1046	581	1026	
	Total green time (s)	4045	11380	4700	11214	
	Throughput (veh)	3075	12601	3167	12382	
	Ave. discharge headway (s)	**2.63**	**1.81**	**2.97**	**1.81**	
Q 3	Ave green time (s)	8.01	15.19	8.14	15.2	35154
	Total phase frequency	433	844	501	867	
	Total green time (s)	3468	12820	4078	13178	
	Throughput (veh)	3347	13844	3718	14245	
	Ave. discharge headway (s)	**2.07**	**1.85**	**2.19**	**1.85**	
Q 4	Ave green time (s)	8.01	10.54	8.04	10.33	33513
	Total phase frequency	535	1013	641	1058	
	Total green time (s)	4285	10677	5154	10929	
	Throughput (veh)	3229	13454	3497	13333	
	Ave. discharge headway (s)	**2.65**	**1.59**	**2.95**	**1.64**	

Table 4.3. Signal timings and average discharge headway for Q-learning in a single intersection

From Table 4.3., the highest throughput was found with Q3, showing that for both Q-learning and ADP, an implementation using an estimate for the time vehicles have spent in the link in the state of the system resulted in improved results.

In terms of signal timings, the through phases were displayed more often than the left-turn phases with a ratio of about 2:1, and in the case of Q3 the duration of the through phase was about double the duration of the left-turn phase. This mimics the actual traffic distribution, with about 20% of the green time dedicated to left-turn phases and the remaining time for through movements. In comparison with ADP, Q-learning phases for the through movements were longer and generated fewer phase changes, and therefore reduced the lost time. The effect of having these longer phases in terms of delay is examined for each movement, as follows.

The average delay for all vehicles in the system is shown in Figure 4.7. for the four Q-learning implementations. At the end of the 160 runs the four implementations seem to converge to the same lower delay level, with faster learning rates for the

algorithms that made use of the time vehicles have spent in the link as part of the state (Q1 and Q3).

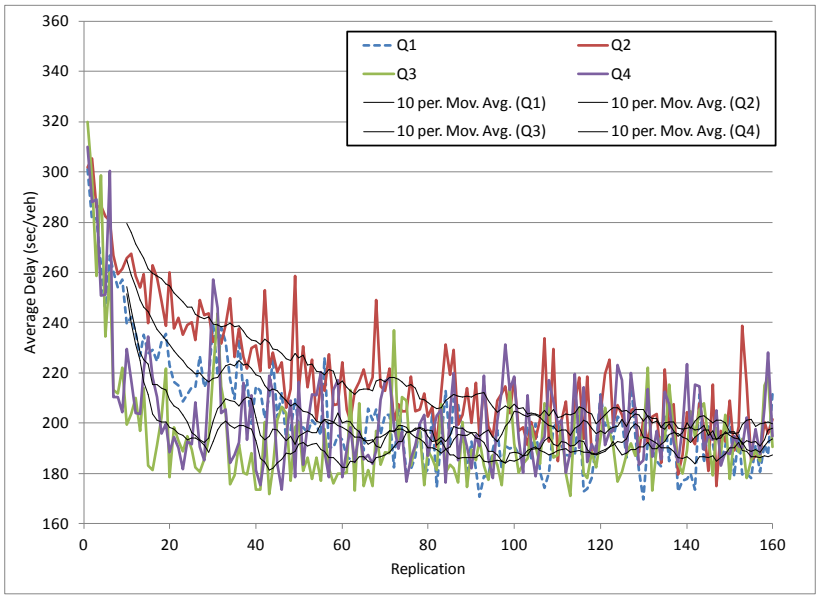


Figure 4.7. Learning curve for average delay of Q-learning algorithms in a single intersection

In comparison with ADP, Figure 4.8. shows the implementations with the lowest delay for both Q-learning and ADP, which in this case were implementations Q3 and ADP 1. It is clear that the performance of the two implementations is similar in terms of delay and this is also reflected in their similar average discharge headway, however they yielded different throughputs (Tables 4.1. and 4.3.).

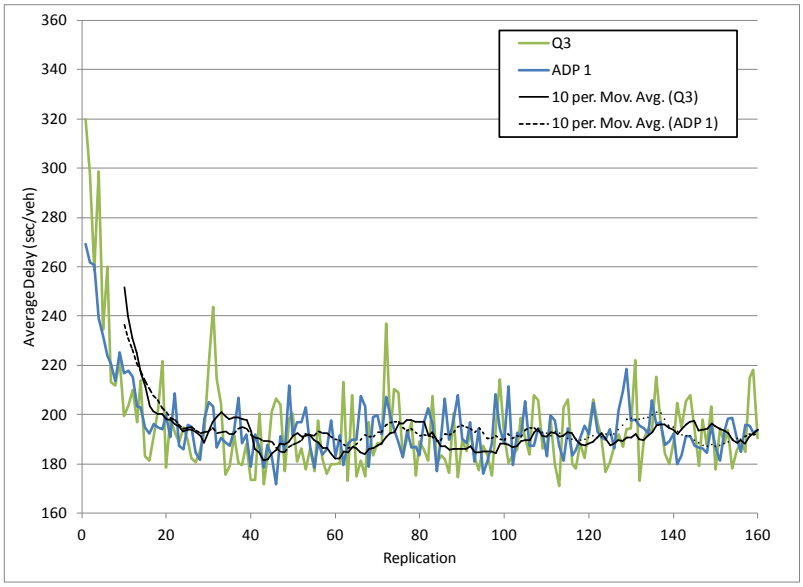


Figure 4.8. Learning curve for average delay of best Q-learning and ADP in a single intersection

In a more detailed examination of the delay of Q-learning, the individual phases are observed to obtain the data shown in Table 4.4., analogous to Table 4.2. for ADP. It is observed that the lowest overall delays were observed for left-turning drivers using Q2, but causing a significant unbalance with delays of through vehicles. The delay of through movements was more predictable, with variances significantly lower than those of left-turn vehicles. Better balance of service for both directions was achieved by Q4 and Q3.

Given that the demand for through movements is 4 times greater than that of left turns, it is not surprising that Q3 had the lowers overall delay for the whole intersection together, as seen in Figure 4.7.

	Left-turn phases		Through phases		Ratio
Implementation	Mean (s)	Variance (s)	Mean (s)	Variance (s)	Left/Through
Q 1	179.2	248.6	256.3	76.0	0.70
Q 2	175.6	77.1	282.7	13.4	0.62
Q 3	244.5	856.2	220.3	33.8	1.11
Q 4	235.0	1443.0	239.5	4.2	0.98

Table 4.4. Delay per phase for Q-learning implementations in a single intersection

Regarding the average speed of vehicles, a summary of the performance of the four Q-learning implementations is shown in Figure 4.9.. Similar to the curve for delay, the speed of the four cases approach a similar speed level by the time they reach the last of the 160 replications in the learning stage.

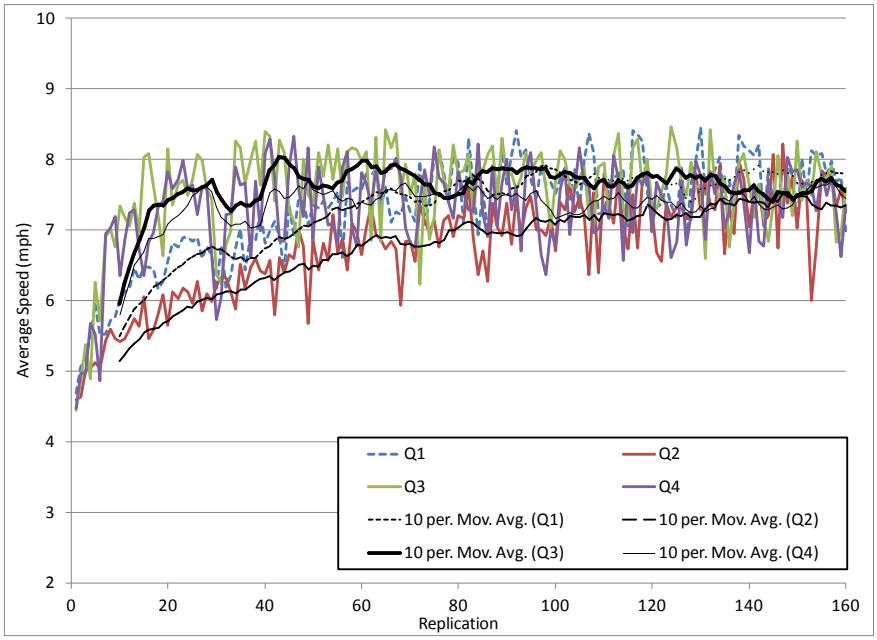


Figure 4.9. Learning curve for average speed of Q-learning algorithms in a single intersection

4.2 Four-intersection Arterial, undersaturated conditions

As mentioned above, the second case study used to evaluate the algorithms was an arterial with four intersections. Conflicting volumes in the first two intersections create the need to continuously change phases, and open the opportunity to observe if there is any emergent coordinated behavior between them. The remaining two intersections do not have conflicting volumes and the signals should learn not to provide green time to those approaches. Entry volumes on the north and south end of the arterial are 2000 vph for the two lanes combined, and one third of the per-lane volume was input

at intersections 1 and 2, for a total of 1000 vph in the three lanes combined. A schematic representation of the arterial is shown in Figure 4.10.

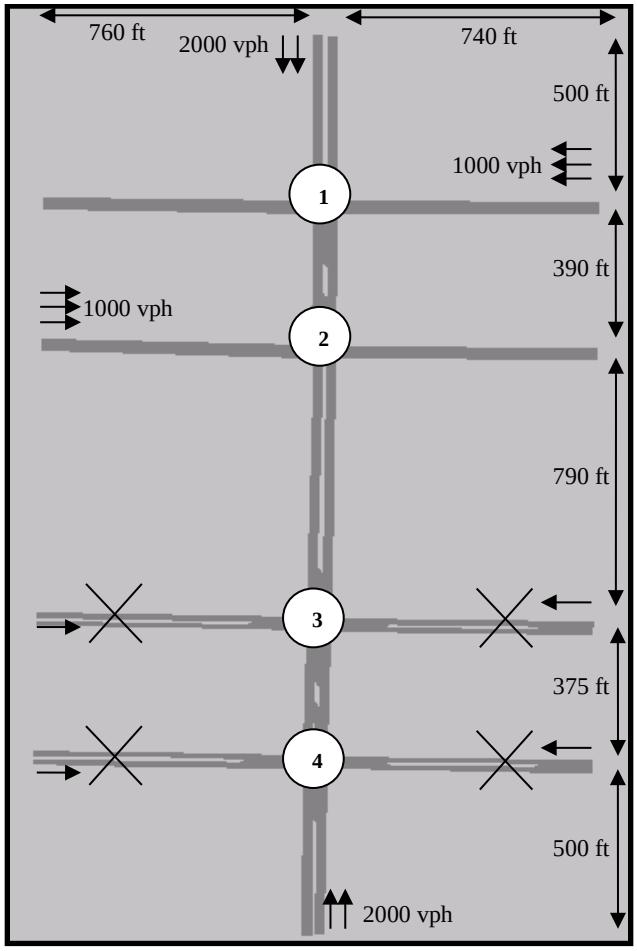


Figure 4.10. Schematic representation of arterial, where "x" indicates no traffic in the approaching links

This section presents the results of multiple implementations, in a similar format to that used for the single intersection case. The analysis of ADP will be described next, followed by the analysis of Q-learning and some contrasts between the two approaches.

4.2.1 ADP Implementations

A total of four implementations were created for this scenario using ADP algorithms. The implementations will be numbered using the letter "*a*" to create a

distinction between this scenario and the others. This will be followed by the next scenarios as well, using letters to prevent confusion between implementations.

The first two implementations (ADP1a and ADP2a) are analogous to implementations ADP1 and ADP3 from the previous scenario (the single intersection). Thus, in ADP1a the state was represented only by the number of vehicles in each link and the current phase, and in ADP2a the state incorporates a measure of the time the vehicles have spent in the link together with the number of vehicles. It is noted that the state space does not change from ADP1a to ADP2a, but only the variables involved in the estimation of the current state.

The remaining two implementations (ADP3a and ADP4a) included the following communication capabilities: 1) it was known to an agent if the receiving links of the neighboring intersections were near capacity (implemented as a dimension in the state), and 2) the agent will receive an incentive for providing green to incoming vehicles from adjacent intersection (implemented as a reduction in penalties). In addition to these capabilities, ADP4a used a modified reward function that included potential downstream blockages, so that penalties were created if green time was given to approaches that could result in these situations. More specifically, penalties were gradually increased if the downstream link was occupied between 0 and 40%, between 40 and 60%, or higher than 60%, as a function of the opposing traffic.

The potential for blockage in ADP3a and ADP4a was included as an additional dimension in the state space in the form of up to two levels of potential blockage per direction. The additional information included in ADP4a did not affect the size of the state space, but the calculation of the reward.

Recall that this scenario was studied in undersaturated conditions, thus performance indicators such as total throughput should be maintained approximately stable for all implementations unless their performance is significantly subpar compared to the others. A similar set of the indicators used in the case of a single intersection will be shown in this case as well, in combination with other indicators that are appropriate for multiple intersections such as the number of stops for the vehicles in the system.

4.2.2 Performance

The analysis begins with the average delay of all vehicles in the network, as shown in Figure 4.11. It is observed that the performance of the four implementations varied significantly, including the last replications of the training curve. ADP3a achieved the lowest average delays and ADP1a the highest. Recall that ADP3a included an incentive for incoming vehicles from adjacent intersections, but so did ADP4a using a different reward function.

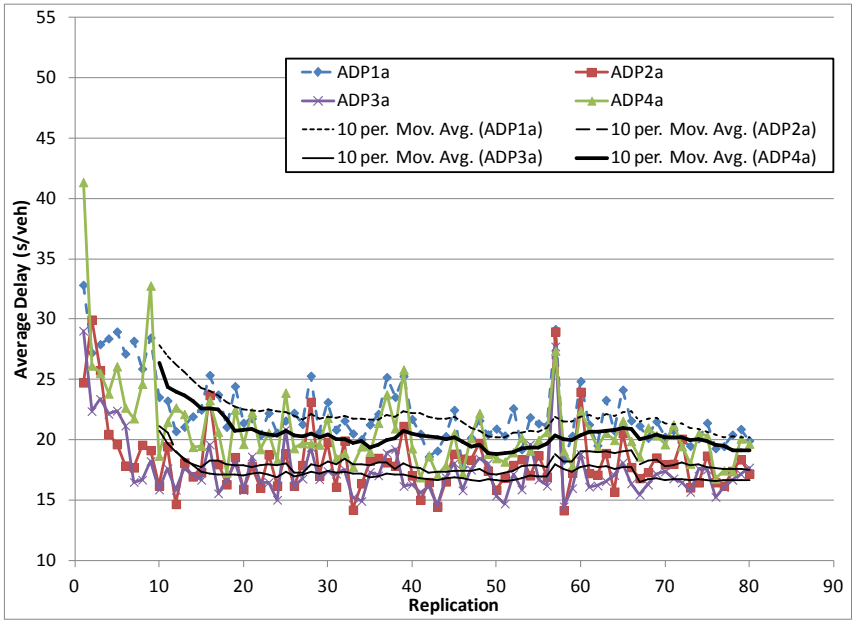


Figure 4.11. Learning curve for average delay of ADP algorithms in an arterial

Differences between the four implementations were, for the most part, the result of not completely eliminating phase changes for the two intersections that did not have conflicting volumes. The signals at these locations were not stable enough using ADP1a and ADP4a and the green phase was at times assigned to the E-W direction. A closer view of the signals in these two intersections showed that ADP1a provided green to the opposite direction for about one fourth of the total green time at intersection 4 and only a negligible portion of the green at intersection 3 (about 1% of the time). On the other hand, using ADP4a about one sixth of the green time was allocated to the E-W direction

at intersection 4, and about 5% to the E-W direction at intersection 3. The remaining two implementations did not provide green time to approaches without demand.

On the other hand, an inspection of the delay was also conducted for each of the phases at the two intersections with conflicting volumes. These results are shown in Table 4.5.

Intersection	Implementation	N-S phase		E-W phase	
		Mean (s)	Variance (s)	Mean (s)	Variance (s)
Intersection 1	ADP1a	15.4	269.6	25.0	2.1
	ADP2a	18.8	225.8	17.3	0.8
	ADP3a	16.3	280.3	23.3	2.4
	ADP4a	16.0	284.7	23.1	2.3
Intersection 2	ADP1a	10.3	81.5	20.7	1.4
	ADP2a	11.7	117.8	18.0	0.6
	ADP3a	11.5	125.6	20.8	2.4
	ADP4a	11.1	116.9	21.5	2.4

Table 4.5. Delay per phase for ADP implementations in an arterial

Overall, from Table 4.5. the average delays between the four implementations are similar, indicating limited effects of the additional features for ADP3a and ADP4a. This may not come as a surprise as the undersaturated conditions were not expected to generate blockages. However, the incentives and provide green times for oncoming traffic could have had an impact in the operation in terms of delay, but this did not seem to be the case.

Additional information about the signal timings for each of the two directions of traffic in these two intersections is shown in Table 4.6.

Implementation	Indicator	Intersection 1			Intersection 2		
		E-W	N-S	Total Througput	E-W	N-S	Total Througput
ADP 1a	Ave green time (s)	8.07	24.89	44314	8.3	20.77	43958
	Total phase frequency	877	876		974	977	
	Total green time (s)	7082	21810		8092	20294	
	Throughput (veh)	9912	34402		9994	33964	
	Ave. discharge headway (s)	**2.14**	**2.54**		**2.43**	**2.39**	
ADP 2a	Ave green time (s)	8.01	30.42	44294	8.04	21.67	43949
	Total phase frequency	769	777		952	956	
	Total green time (s)	6164	23640		7654	20716	
	Throughput (veh)	9919	34375		9939	34010	
	Ave. discharge headway (s)	**1.86**	**2.75**		**2.31**	**2.44**	
ADP 3a	Ave green time (s)	8.07	26.57	44323	8.06	22.71	43947
	Total phase frequency	850	849		930	931	
	Total green time (s)	6860	22558		7496	21143	
	Throughput (veh)	10008	34315		9999	33948	
	Ave. discharge headway (s)	**2.06**	**2.63**		**2.25**	**2.49**	
ADP 4a	Ave green	8.11	23.42	44299	8.15	21.52	43989
	Total phase frequency	907	908		953	953	
	Total green time (s)	7360	21270		7766	20508	
	Throughput (veh)	9907	34392		9956	34033	
	Ave. discharge headway (s)	**2.23**	**2.47**		**2.34**	**2.41**	

Table 4.6. Signal timings and average discharge headway for ADP in arterial

From Table 4.6., the signal timings from ADP2a and ADP3a provided longer green times for the traffic direction along the arterial compared to the other implementations. Likewise, the average discharge headway was slightly longer for ADP2a and ADP3a.

In addition, the average green times for intersection 1 were longer than for intersection 2 along the arterial. This is explained by the more continuous arrival of vehicles at intersection 1 given that the demand on the northbound is reduced to about 72% of the original entry volume due to right and left turn movements, and the demand southbound to about 95%. This is also an indication that, given the greater demand

southbound, coordination should have been provided in this direction. The offsets between the beginning of green time at intersections 1 and 2 on the southbound and on the northbound were explored to determine if the coordination occurred as expected.

For the southbound, the offsets of the last 20 replications (at the end of the training period) were found to be shorter than those for northbound and closer to an ideal offset given the distance between intersections. The ideal offset assuming no initial queue was around 10 seconds in free-flow speed, but closer to 15 seconds with the assigned demands. For example, a plot of the cumulative distribution of the offsets using ADP3a showed that 70% of the offsets in the southbound direction were lower than 22 seconds, whereas in the southbound the 70% of the cumulative distribution was located at 34 seconds. This is a clear indication of better coordination in the southbound, as expected. Another example without the coordination features (using ADP2a) showed that 70% of the offsets were slightly longer in both directions, with the southbound direction at 24 seconds and for the northbound at 38 seconds.

Another measure of the coordination of traffic along the arterial is the average number of stops per vehicles. Even though this indicator does not account for vehicles slowing down, it may show when coordination was significantly different between the implementations. This is shown in Figure 4.12, where ADP2a and ADP3a, following the same trend observed for the delay, and closely linked to the signal operation in intersections 3 and 4.

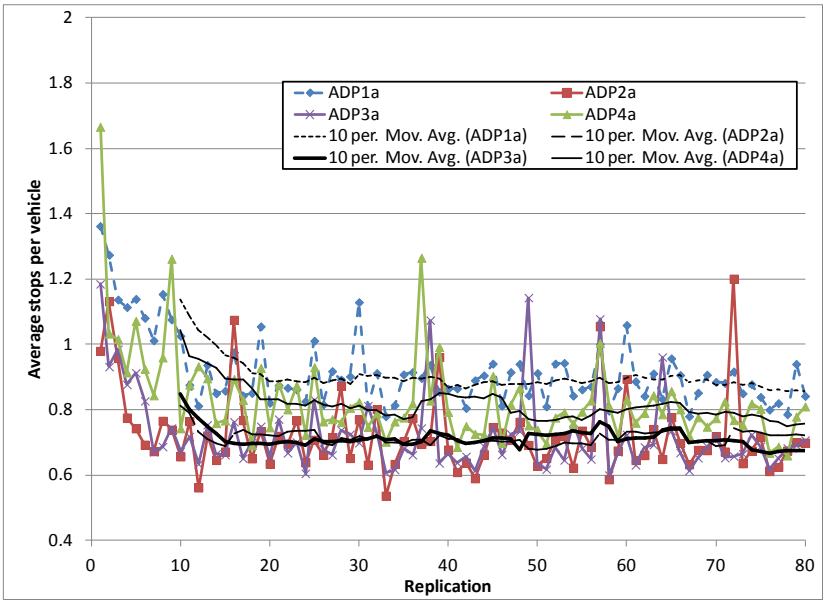


Figure 4.12. Learning curve for average number of stops of ADP algorithms in an arterial

4.2.3 Q-learning Implementations

Four implementations similar to those explained above for ADP where used to test the performance of Q-learning in the arterial scenario, named Q1a through Q4a. There is correspondence between the labeling used in this subsection and the characteristics of the implementations for ADP, thus for example the implementation for Q1a had the same state and reward definitions of ADP1a.

However, unlike the results for ADP all four cases using Q-learning had similar performance at the end of the 80 training runs and reached the same levels of the best ADP cases.

4.2.4 Performance

The first indicator used in this analysis was the average delay per vehicle in the system, as shown in Figure 4.13. The four Q-learning implementations converged to a similar delay value and produced similar variations on the replications. An examination of the delays per intersection showed that in one of the implementations (Q1a) the signals provided momentarily the right of way to the approaches with no demand, delaying vehicles unnecessarily. In Q1a the signals provided on average about 5% of the total

green time to the approaches with no demand, which accounts for some of the increased total delay of Q1a compared to the other algorithms in Figure 4.13. Additional reinforcement from adding an estimate of delay in the state, as well as incentive from adjacent intersections had a better effect in preventing switching phases upon no demand in the Q-learning implementations compared to ADP.

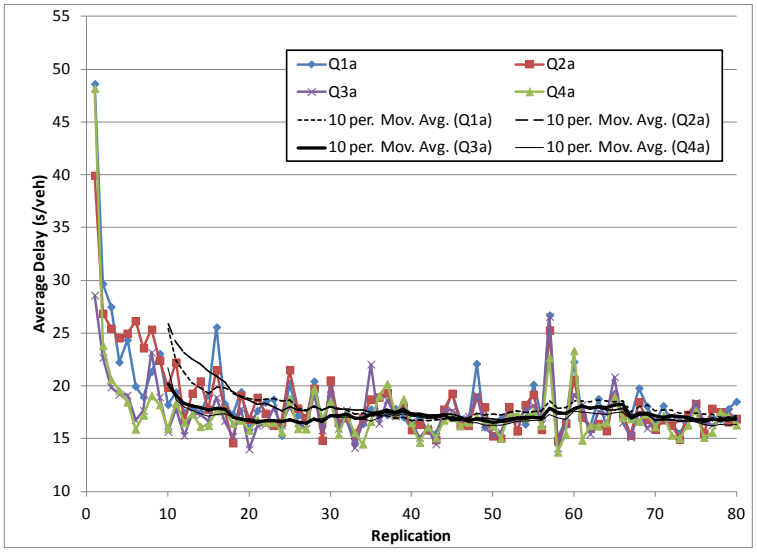


Figure 4.13. Learning curve for average delay of Q-learning algorithms in an arterial

Delay values for the two intersections with opposing demands are shown in Table 4.7. Delays for the N-S direction were in general lower than for the E-W direction, which may be at first counterintuitive given the greater demand on the N-S direction, but it can be mainly explained by the greater number of vehicles that could be processed without stopping due to increased pressure to hold the green light. The larger variance of the delay for the N-S direction also explains this situation, where some vehicles may have been processed by the intersection without stopping but some others had to wait at least the minimum green time and yellow-red transition of the E-W direction. On the other hand, vehicles in the E-W direction were likely to wait for the duration of the N-S direction (a great portion of a typical cycle) to be processed in the next green light, having a more constant delay. Lastly, a slight decrease in the delay of the intersections on the N-S direction (along the arterial) can be observed when using Q4a (which accounted

for incentive upon arrival of platoons) but at the expense of greater delay for the E-W direction.

Similar comments to those mentioned for the ADP implementations apply to these cases with Q-learning in relation to the magnitude of the mean and variance of the delay.

Intersection	Implementation	N-S phase		E-W phase	
		Mean (s)	Variance (s)	Mean (s)	Variance (s)
Intersection 1	Q1a	17.1	267.4	21.4	1.2
	Q2a	17.1	208.1	17.6	1.0
	Q3a	17.5	284.6	19.6	1.7
	Q4a	16.0	284.7	23.1	2.3
Intersection 2	Q1a	13.3	156.4	15.5	0.4
	Q2a	13.4	150.5	15.2	0.7
	Q3a	12.3	136.3	18.1	1.1
	Q4a	11.1	116.9	21.5	2.4

Table 4.7. Delay per phase for Q-learning implementations in an arterial

The characteristics of the signal timings and the average discharge headway for the four implementations are shown in Table 4.8. As expected, given the undersaturated conditions, all four algorithms processed a very similar number of vehicles. Slightly different discharge headways were observed using Q3a and Q4a compared to Q1a and Q2a, favoring the N-S direction (larger headways) and also signal progression.

Implementation	Indicator	Intersection 1			Intersection 2		
		E-W	N-S	Total Througput	E-W	N-S	Total Througput
Q1a	Ave green time (s)	8.14	23.47	44304	8.16	20.73	43952
	Total phase frequency	906	904		979	977	
	Total green time (s)	7375	21217		7989	20253	
	Throughput (veh)	9957	34347		10013	33939	
	Ave. discharge headway (s)	**2.22**	**2.47**		**2.39**	**2.39**	
Q2a	Ave green time (s)	8.23	21.17	44333	8.24	19.63	44019
	Total phase frequency	962	967		1008	1003	
	Total green time (s)	7917	20471		8306	19689	
	Throughput (veh)	9929	34404		10026	33993	
	Ave. discharge headway (s)	**2.39**	**2.38**		**2.49**	**2.32**	
Q3a	Ave green time (s)	8.28	23.05	44250	8.46	22.05	44006
	Total phase frequency	912	917		932	935	
	Total green time (s)	7551	21137		7885	20617	
	Throughput (veh)	9922	34328		9980	34026	
	Ave. discharge headway (s)	**2.28**	**2.46**		**2.37**	**2.42**	
Q4a	Ave green time (s)	8.32	26.7	44309	8.48	26.56	43966
	Total phase frequency	841	838		834	833	
	Total green time (s)	6997	22375		7072	22124	
	Throughput (veh)	9968	34341		9992	33974	
	Ave. discharge headway (s)	**2.11**	**2.61**		**2.12**	**2.60**	

Table 4.8. Signal timings and average discharge headway for Q-learning in arterial

Different from ADP, the average phase duration for both N-S and E-W directions are more similar between intersections 1 and 2, creating a better probability of coordination in both directions due to common cycle length. If this is true, the offsets in both directions should be similar to each other. Therefore, an examination of the offsets for the implementation of Q4a was examined to determine the similarity of the offsets. Results showed a closer agreement between the two distributions, with the 70% of them being 22 seconds or lower for the N-S direction and 26 seconds or lower for the E-W direction. A sample image of the two distributions is shown in Figure 4.14, and indicates that the offsets varied in a very similar way throughout the 20 last replications, favoring coordination in the two directions of traffic.

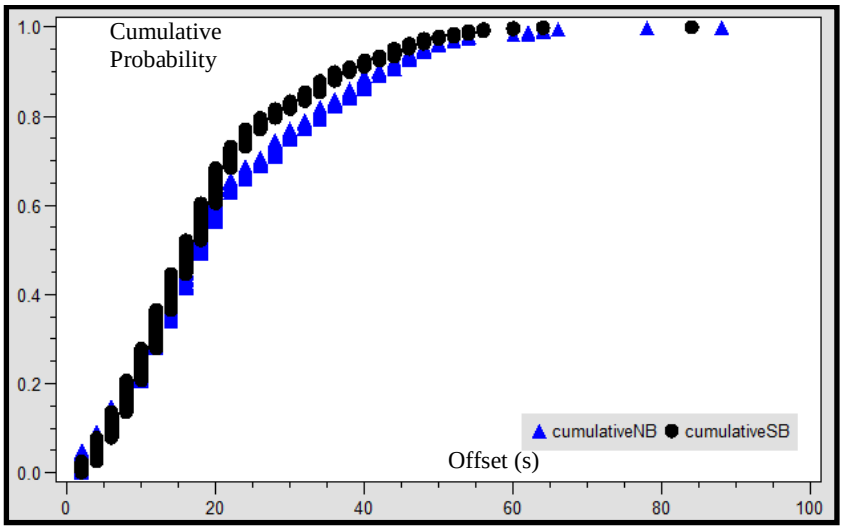


Figure 4.14. Cumulative distribution of offset durations for NB and SB in intersections 1 and 2 using Q4a

The total number of stops per vehicle was also monitored for the whole system, and it is shown in Figure 4.15. The four implementations converged to a value of about 0.7 stops per vehicle, with an edge for the Q4a implementation. This was also expected given the longer average green time for the N-S direction in Q4a compared to the other implementations and the similar timings for the two intersections with conflicting movements, as shown above.

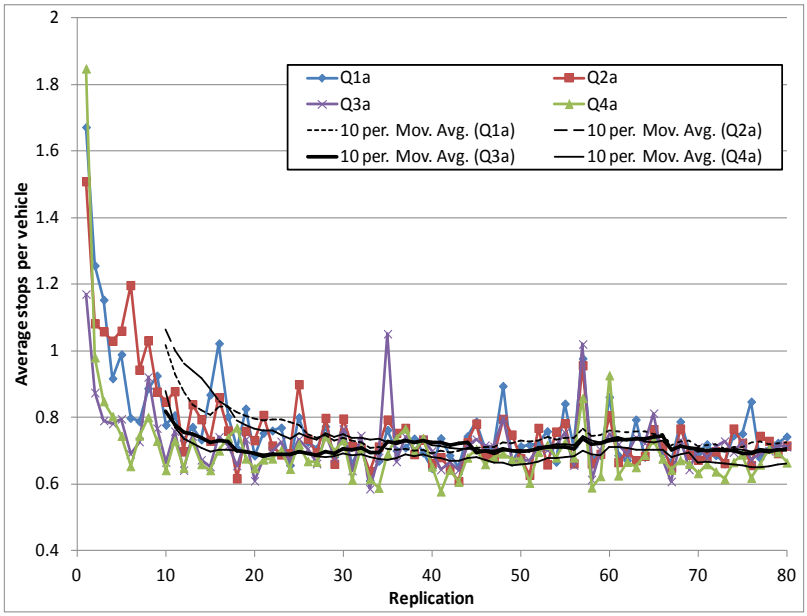


Figure 4.15. Learning curve for average number of stops of Q-learning algorithms in arterial

In comparison with the best ADP, the learning curve of the Q-learning implementation was very similar, with slight benefits in terms of the number of stops for Q-learning. This can be observed in Figure 4.16, showing ADP3s and Q4a.

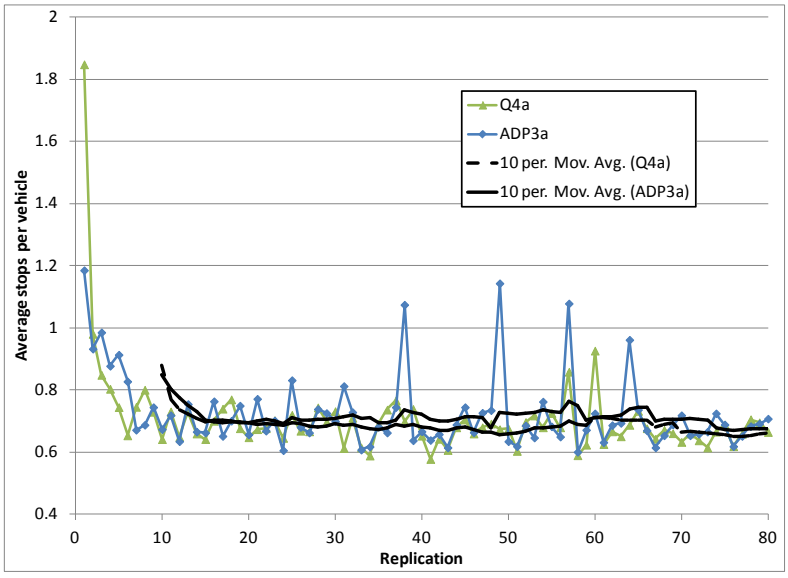
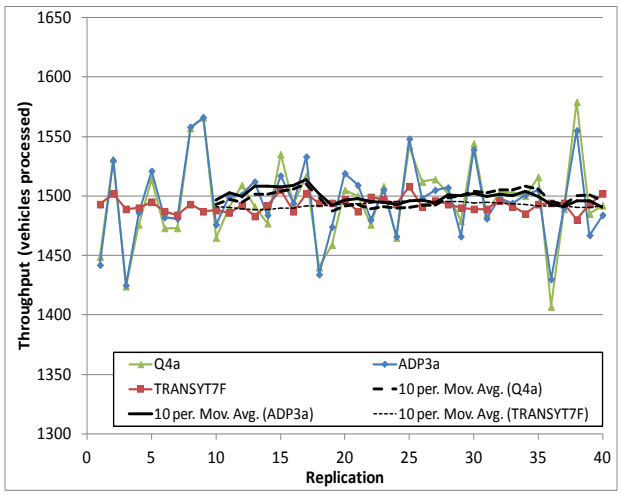


Figure 4.16. Comparison of learning curves for average number of stops of Q4a and ADP3a

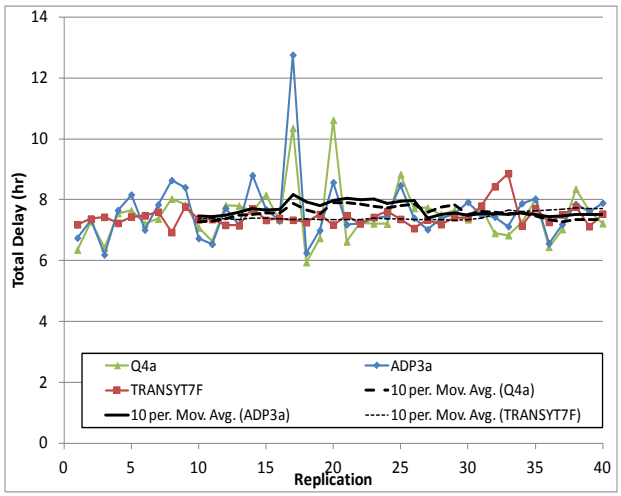
The performance of the best ADP and Q-learning algorithms was also compared to the results of the traffic signal optimization performed by the commercial software package TRANSYT7F, which uses a search in the solution space through a genetic algorithm. The traffic environment for TRANSYT7F was provided by CORSIM, a well known microscopic simulator.

The arterial was coded in CORSIM with the exact same characteristics as in VISSIM. In addition, calibration had to be performed to ensure that the vehicle characteristics, the discharge headways and speeds were the same in the two simulation environments. The following variables were modified in VISSIM to attain the desired calibration: desired speed, vehicle types were limited to two, with the same dimensions and similar operational characteristics, the additive part of the desired safety distance in the car-following model (to obtain similar discharge headways), and the standstill distance of vehicles (to match the number of vehicles that a link could store). It is noted that the decision to perform this comparison was made before obtaining VISSIM results presented in this report; therefore all data presented up to this point and onward was obtained in VISSIM after this calibration was performed.

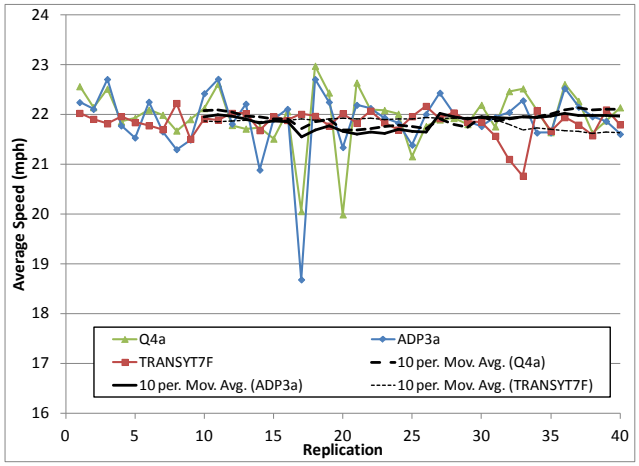
The comparison of ADP and Q-learning with TRANSYT7F was performed in terms of average delay per vehicle, average vehicle speeds, and total system throughput. The last 40 replications of the training for ADP and Q-learning were used in the comparison whereas 40 replications were obtained from CORSIM using the signal timing settings after the optimization process was completed. Results of the comparisons are shown below in Figure 4.17.



19a – Throughput



19b – Total vehicle delay



19c – Average vehicle speed

Figure 4.17. Comparison performance of Q4a, ADP3a and TRANSYT7F in undersaturated arterial

Figure 4.17. shows similar average values for all three methods for the three indicators. However, higher variation between different replications was obtained in VISSIM compared to CORSIM. It is important to observed that while the same random seeds where used for ADP and Q-learning, this was not possible with CORSIM, as the simulation packages had a different car following model, and therefore different use of random numbers. This variation can be better observed in 19a, where the vehicle throughput is shown for the different replications.

These results indicate that the ADP and Q-learning implementations were as effective as current commercial solutions in finding the signal timings of the arterial studied in this study, with undersaturated conditions. Results can also be seen as a building block for more complex scenarios described in the following subsections.

4.3 5x2 Network, Undersaturated Conditions

The third scenario included in this study was a small network of ten intersections in a 5x2 configuration. As described above, there were single-lane links along the 5 contiguous intersections in the E-W direction, and a combination of 3-lane and 2-lane

intersecting streets on the N-S direction. This particular set of experiments had demands that were slightly below saturation, with higher inputs per lane in the E-W direction. All entry links in the E-W direction received 1000 vphpl whereas all links in the N-S direction received one third of this volume per lane (333 vphpl). A schematic representation of the network is shown in Figure 4.18.

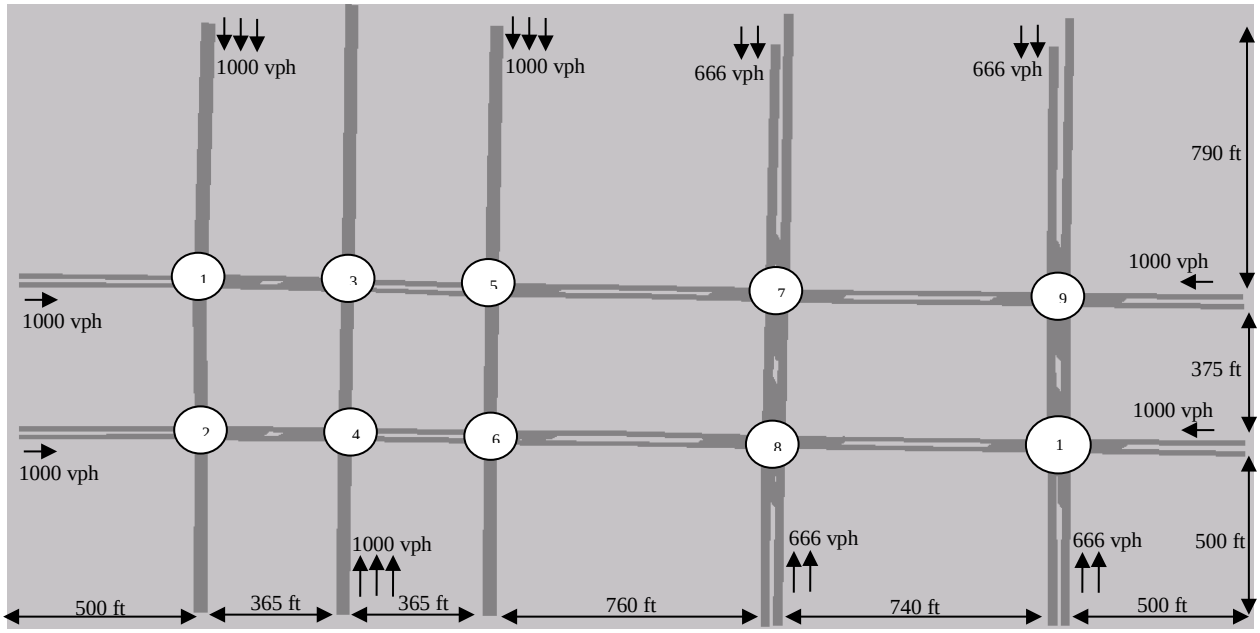


Figure 4.18. Schematic representation of 5x2 network

4.3.1 Implementations

Similar to the previous two scenarios, a set of implementations were tested to determine their performance. The following are the descriptions of the implementations, which include either and ADP approach or a Q-learning approach:

- ADP1b: The definition of the state for this implementation includes a component for each direction of traffic that is estimated using both the number of vehicles and the time they have already spent in the link. This is a similar implementation to that used in previous scenarios, such as ADP3 and ADP2a.

- ADP2b: This implementation included a factor to account for potential blockages due to downstream congestion. This factor was represented in the state as an additional dimension, thus one dimension for each direction was created. This factor also affected the rewards by increasing the relative weight of the link without potential blockage, therefore favoring the green light in that direction. The reward is analogous to that used in ADP4a.

- Q1b: In this case, an application using Q-learning was created not only including the blockage factor from ADP2b, but also some incentives for anticipating vehicles from adjacent intersections. This incentive was in the form of added weight to the direction expecting the vehicles. Even though this feature is expected to produce better results with very low traffic in one of the traffic direction, it was included in this scenario to determine if it had any impact in the network.

- Q2b: This implementation had the same state definition as Q1b, but the calculation of the rewards was estimated using the same definition from ADP4a, therefore the blockages and incentives have a significant impact in the rewards for each action.

4.3.2 Performance

The performance of the implementations is analyzed next, beginning with the delay for all vehicles in the network as the agents trained (shown in Figure 4.19). An improvement in the average delay of all vehicles in the network is observed as the agents trained. It is noted, however, that the change in performance between the initial portion of the training and the last of the replications was in the order of 10% or less.

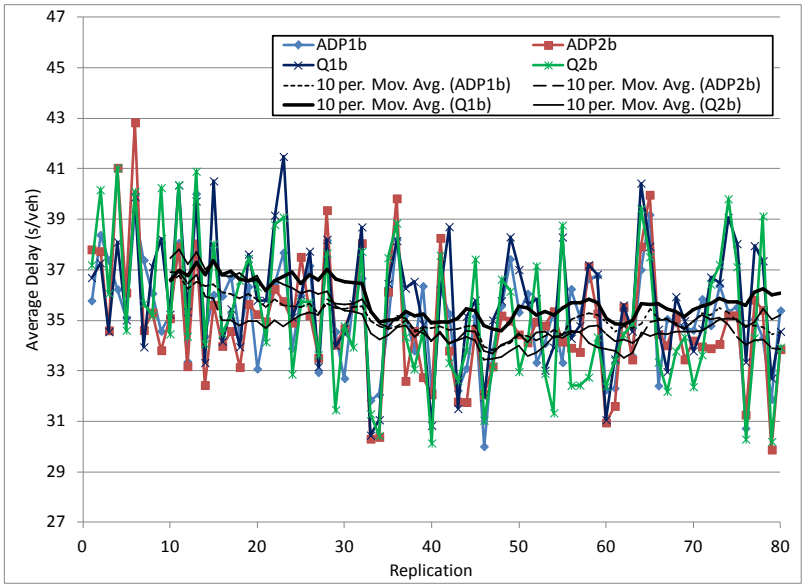


Figure 4.19. Learning curve for average delay of RL algorithms in 2x5 network

An analysis of the queue lengths in all links in the network, and for all implementations, showed that the only points that eventually had queues near their capacity (>85%) were left-turn lanes and the eastbound link of intersection 4. Therefore the signals prevented queue spillbacks on the through movements but due to the permitted operation of the left-turns (as opposed to using an exclusive phase), these eventually created queues that reached the main line. Given that only a few links were likely to be blocked, it is not surprising that the total throughput of the network was similar for all implementations and fluctuated around the expected number of vehicles to be processed in each of the 15-minute replications, which in this scenario was around 2400 vehicles (Figure 4.20).

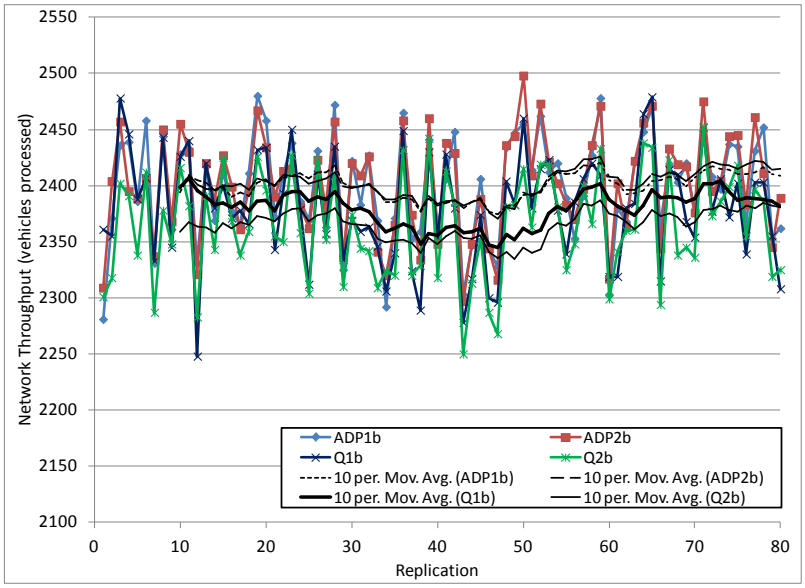


Figure 4.20. Learning curve for network throughput of RL algorithms in 2x5 network

Then, the actual signal timings were examined to determine how green times were utilized at each intersection. The direction of traffic with greatest volume was monitored in detail (E-W) since the main problematic areas were observed along these links. In addition, the discharge of left-turning vehicles was more critical on the E-W links given that there was only one through lane and it could be easily blocked by left-turning vehicles overflowing the turning lane.

The percentage of green time given to the E-W direction for all intersections was between 57% and 74% of the total green time. Based on the total demand per link, and assuming the same number of lanes for all approaches, the proportion of green time given to the E-W direction should have been about 50% for intersections 1 to 6, and about 66% for intersections 7 to 10. However, given that there is a single through lane on the E-W direction, it is necessary to give additional green time in order to process the same number of vehicles. Therefore, there is a tradeoff between two objectives in the network: providing equal service rates for the two directions of traffic and processing more vehicles per unit of time.

For the network to be more efficient, it is preferred to provide green time to the approaches with greater number of lanes (e.g. the N-S direction), as more vehicles will be

processed per unit of time. However, approaches in the E-W can develop long queues and this may result in eventual blockages, even during the green phase in the N-S since there are incoming vehicles to the E-W links from right- and left-turning movements.

From the analysis of the signal timings, it was observed that the lowest and highest ratios of green times for the E-W direction were located at intersections 2 and 4, respectively. This explains the relatively long queues found in the eastbound direction of intersection 4, as mentioned above. Incoming vehicles in the eastbound direction entered the link using 74% of the green time at intersection 2, but only had 57% of the green at intersection 4 to be processed.

Lastly, the average speed of vehicles in the network improved also in a similar proportion than the delay during the training period (see Figure 4.21). It is noted that the improvements in the system as training progresses should be observed by looking at the throughput, delay, and speed simultaneously. In this case delay decreased and speed increased while maintaining constant throughput (which was equal to the total demand), but in oversaturation delays may increase and speed decrease while the throughput is improved. This situation is examined in the next subsection.

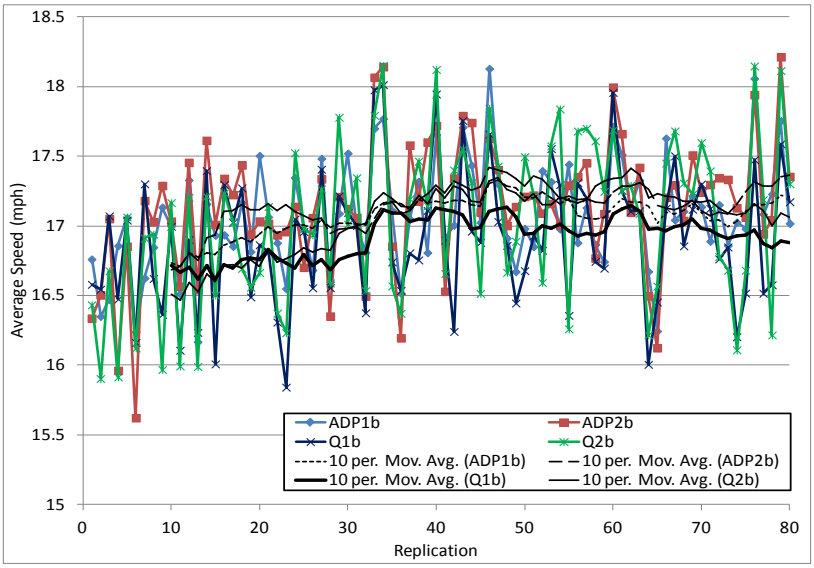


Figure 4.21. Learning curve for average vehicle speed of RL algorithms in 2x5 network

4.4 5x2 Network, Oversaturated Conditions

This scenario was evaluated to provide an indication of the performance of the RL algorithms in oversaturated conditions. As opposed to single intersections, where oversaturation may create long queues but without preventing the intersection to service vehicles at the front of the queue, in a network the occurrence of gridlocks can completely prevent the intersections to discharge vehicles, collapsing the system without recovery for the agents to learn improved strategies.

The same network used in the previous subsection is used here, but with additional demands in the N-S direction. In total, entry volumes were modified such that there were 1000 vphpl at all entry points, thus for example in a three-lane approach the total entry volume was 3000 vph. These inputs ensured oversaturation and increased the complexity of the scenario. For the traffic signals, a key issue is to prevent blockages in the inner links of the network due to queue spillbacks.

Two algorithms were tested under this condition to illustrate the need of communication between neighboring intersections. The first implementation used Q-learning without communication between intersections or any other form to identify potential blockages downstream, and it was called Q1c. The second implementation was called C2c. It allowed communication between neighboring agents and added the potential for blockages to the state and reward representation, similar to implementations described in the above subsections such as Q2b.

The analysis is focused on the total network throughput and queues rather than speed or number of stops, given the oversaturated conditions of this case. Results of the learning curves for the total network throughput are shown in Figure 4.22., where it is observed that the performance of the agents without communication is clearly lower than those with communication. The number of vehicles processed with communications reached an average of 3870 vehicles processed, which is about 74% of the total number of vehicles that represent the demand at all entry links. A more realistic measure of the

efficiency of the network, however, it is necessary to recognize the existence of signals and therefore a lost time for the yellow-red transitions.

As a rough estimate, the demand per intersection is 2000 vphpl and if it is assumed that about 1600 vphpl can be processed (with an average discharge headway of 2 seconds and subtracting about 10% of lost time), the capacity should be about 1600 vph per exiting lane, or 80% of the total demand. The total number of vehicles trying to enter the network is 5250, therefore the capacity should be about 5250*0.8=4200. If this is the case, the network is currently operating at over 90% efficiency in terms of throughput using the agents with communication.

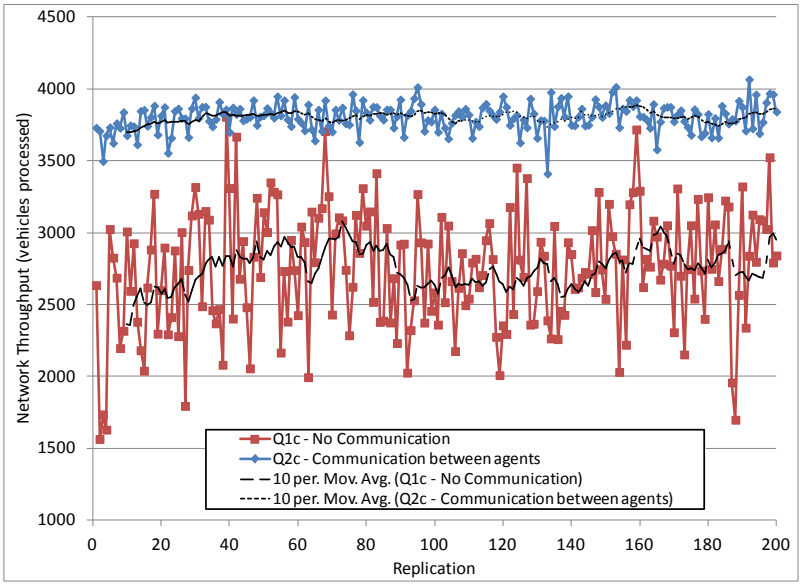


Figure 4.22. Learning curve for network throughput of RL algorithms with and without communication in oversaturated 2x5 network

An examination of the performance of both algorithms showed that the main concern in this scenario was downstream blockages and gridlocks, often created by turning vehicles. Without communication, intersections will strive to process as many vehicles as possible disregarding the available capacity of the receiving links, increasing the potential of gridlocks.

Lastly, the average delay per vehicle for the same implementations is shown in Figure 4.23. As expected delays without communication were significantly higher and did not reach the lower levels as with communication. Also, it is noted that for both algorithms, as the learning occurred, throughput had a tendency to increase while the average vehicle delay decreased.

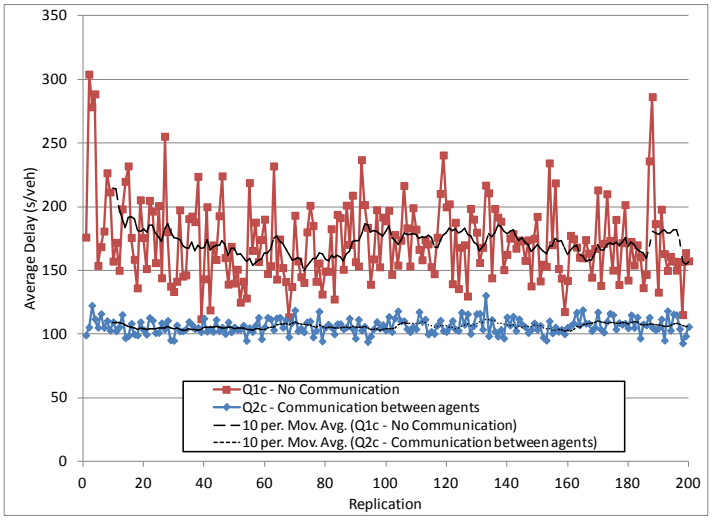


Figure 4.23. Learning curve for average delay of RL algorithms with and without communication in oversaturated 2x5 network

4.5 4x5 Network, Oversaturated Conditions

In this section, a more challenging scenario is used to test the Q-learning and ADP algorithms. A portion of downtown Springfield, IL, was coded in VISSIM for this purpose. Vehicle demands ensured oversaturation in all directions, with 1000 vphpl in all directions, and there is a combination of one-way and two-way streets as well as different number of lanes. Left-turn movements are completed from left-turn pockets that have very limited capacity and tend to block through movements given the oversaturation conditions. Also, the left-turn movements do not have an exclusive phase, but are allowed upon traffic gaps in the oncoming traffic flow. In fact, this network encompasses the two previous scenarios - the arterial with four intersecting streets, and the 2x5 network from

the section above - and expands on their boundaries to form a 4x5 network. A schematic representation of the network is shown in Figure 4.24.

One implementation using ADP and one using Q-learning were tested in this network. The implementations included features to identify potential blockages and to promote flow of incoming platoons from adjacent intersections. These features are similar to those used in previous scenarios. It is highlighted that without elements to identify blockages, the implementations may not be able to learn since the system will not evolve past gridlocks, as it was pointed out in a previous scenario.

An additional implementation using the max-plus algorithm described in the Methodology section was also tested in this scenario. The max-plus algorithm identifies potential actions that may favor coordination based on the occupation level of all links. This is done by quantifying the benefits of selecting the phase in the E-W and the N-S direction for each intersection, and finding an optimal solution for the whole network.

There are a number of ways to implement these benefits in the reward and/or in the state representation. As described in the brief literature review provided in Chapter 2, previous studies have used the results of the max-plus algorithm as the major factor to select the phases in a traffic network and it has not been incorporated within other methodologies such as learning algorithms.

In this study, it was decided to incorporate the results of the max-plus algorithm as a factor to the standard reward values obtained for the E-W and N-S actions. This was implemented by finding the ratio between the max-plus benefits of E-W and N-S and this value is applied as a multiplication factor to the cost of taking one of the two actions. For example, if the max-plus benefit of selecting E-W is measured as 10 and the benefit of selecting N-S is 7.5, then the value of E-W is increased by a factor of 10/7.5=1.33, and the value of N-S is not modified.

Using this procedure the max-plus results can be combined with RL algorithms, and it is possible to bias the agent actions towards improved coordination, in combination with other elements of the reward structure, such as potential blockages.

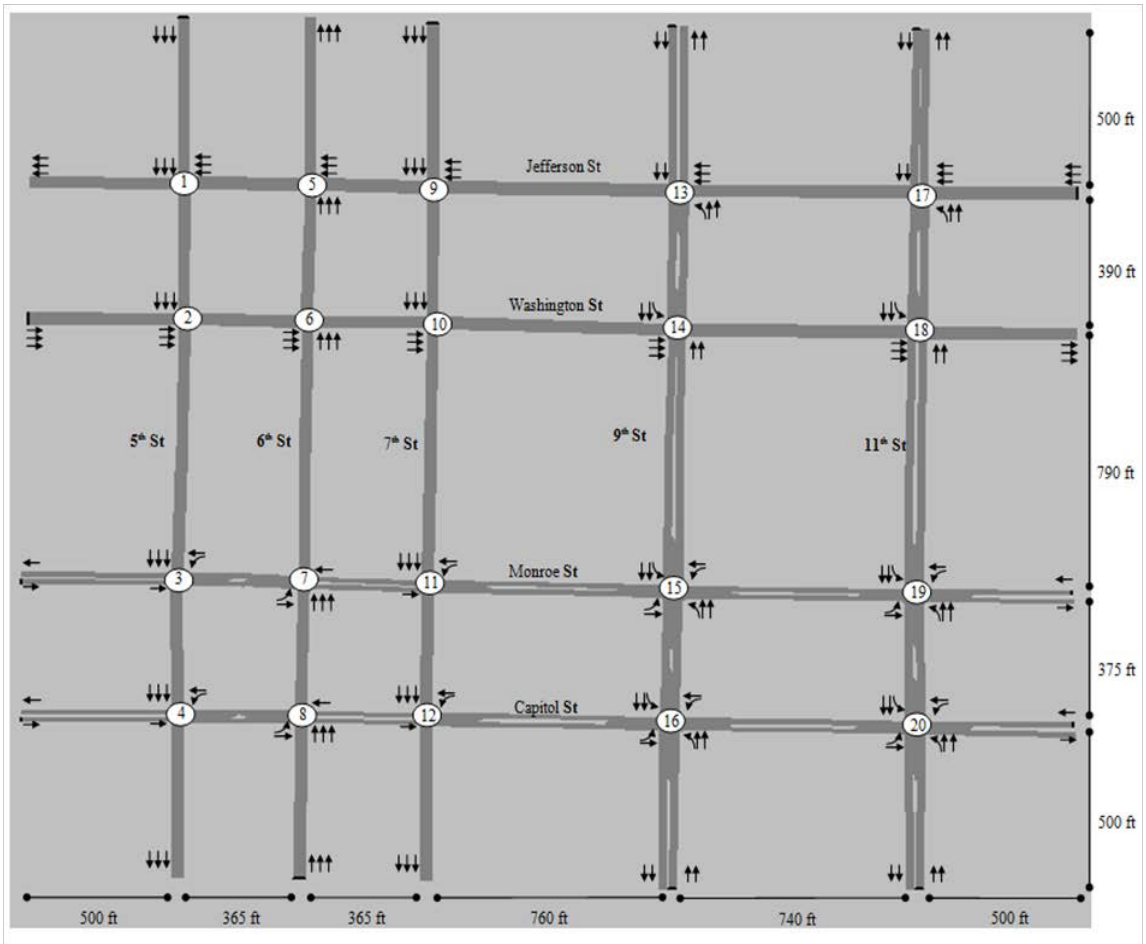


Figure 4.24. Schematic representation of 5x4 network

Similar to the scenario with an arterial in Section 4.2, results from this scenario are compared to TRANSYT7F through the use of similar measures of performance, providing a valid and commercially available reference point. It is noted that as described in Section 4.2., calibration efforts were conducted in order to have meaningful comparisons between results from the RL algorithms in VISSIM and results from TRANSYT7F using CORSIM.

The total network throughput for the four algorithms is shown in Figure 4.25. It can be observed that by the end of the 60th replication all four implementations have

reached a similar number of vehicles that can be processed in the whole network. A series of observations can be drawn from Figure 4.25 as follows:

- Even though the RL algorithms and TRANSYT7F use mechanisms that are very different, they show similar performance in terms of throughput.

- There is a sudden improvement in the performance of ADP past the initial training period, reaching a point where it is comparable to the other strategies. This occurred when the action selection changed from the Boltzman distribution to an e-greedy strategy. The action selection changed when a given state has been experienced "enough" times (in this case each action at least 5 times), so that a more robust estimate of the value of the states exists. Also, e-greedy strategies were used exclusively in the operational mode of the agents, leaving the Boltzman distribution for the training periods.

- The addition of the max-plus algorithm had marginal improvements in the learning at the beginning of the learning stages, but ultimately converged to similar values by the end of the last replication. This could be case due to the nature of the current max-plus implementation itself, where multidirectional coordination may result between neighboring intersections, promoting immediate localized benefits but not necessarily network-wide improvements at the end of the time period.

- Fluctuation in the performance in the last replications is in the order of 10% depending on the initial random seed. This was true for all solutions including TRANSYT7F.

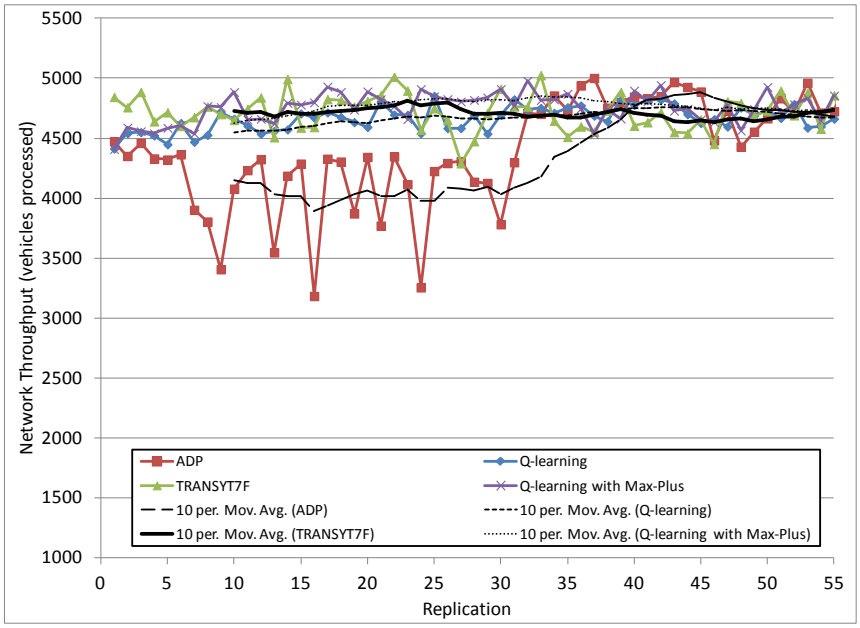


Figure 4.25. Learning curve for network throughput of RL algorithms and TRANSYT7F in oversaturated 4x5 network

Figure 4.26 shows the total delay of vehicles in the network for the 15-minute analysis period. Delay levels seem to be similar for the three implementations using ADP or Q-learning and somewhat larger for the implementation from TRANSYT7F. In addition of the delay, the total number of vehicles in the network was observed to determine if delay for the four implementations could be compared. On average, the total number of vehicles in the network was very similar for the different implementations: 1433 for Q learning, 1511 for Q-learning with max-plus, 1534 for ADP, and 1565 for TRANSYT7F; this indicates that approximately the same number of vehicles was included in the calculation of the total delay.

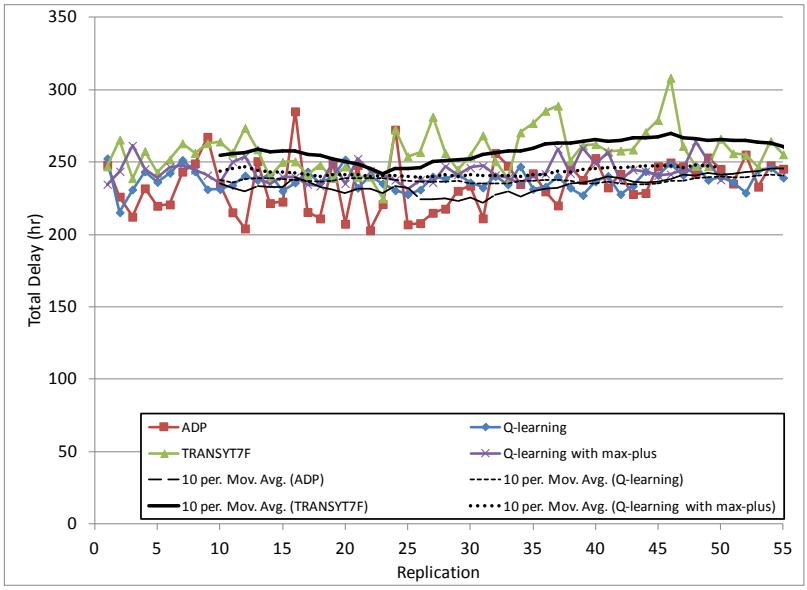


Figure 4.26. Learning curve for total vehicle delay of RL algorithms and TRANSYT7F in oversaturated 4x5 network

More detailed analysis of the max-plus algorithm was conducted in terms of the combination of total network throughput and the average number of stops per vehicle, as seen in Figure 4.27. This rather particular view of the effects of the max plus algorithm in the Q-learning implementation, it is seen that there is a tendency for the average number of vehicles processed to be higher and the number of stops to be lower when the algorithm is added. This is desirable and shows that there could be improvements in performance by incorporating an active coordinating strategy in the learning mechanism.

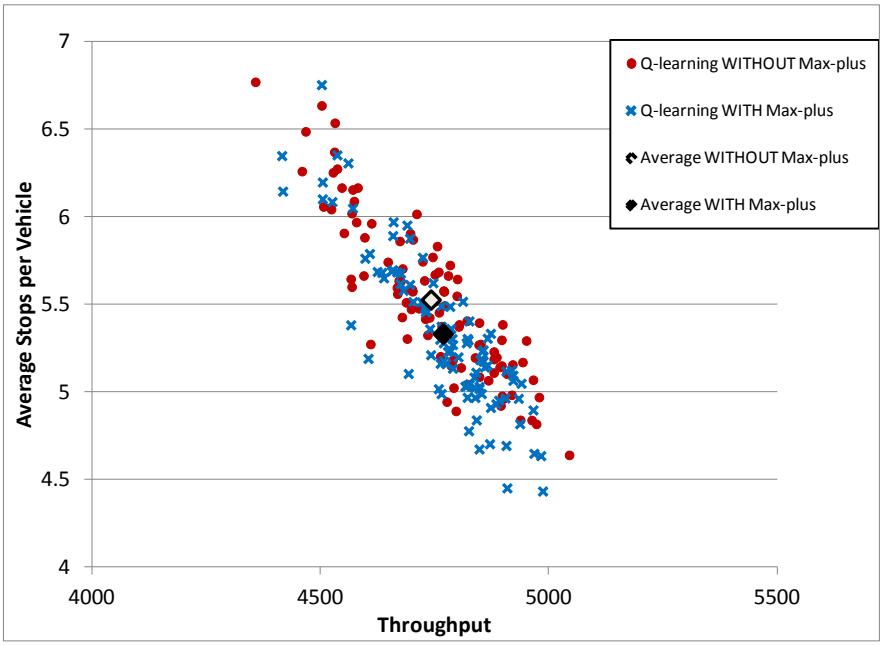


Figure 4.27. Effects of Max-plus algorithm on the Q-learning implementation in the oversaturated 4x5 network

In addition, an evaluation of the queue levels inside the network was conducted to determine the most likely locations of blockages and gridlocks. Using the last 10 replications from both the ADP and Q-learning implementations, it could be observed that all of the queue backups occurred on the lower portion of the network, this is, where a single lane encounter two or three intersecting lanes. Note that this is area is the same used for the 2x5 network in the previous case.

Most of the movements where queue exceeded the capacity were left-turns, as shown in Figure 4.28. In Figure 4.28., short red lines and long black lines show areas of queue overflow for the left-turn pockets and through links, respectively. It is also observed that even though the upper corridors carried the highest volumes per link because they had two and three lanes, queues did not create significant blockages. The issue of having permitted left-turn movements from a pocket with very limited capacity and a single through lane in oversaturated conditions seemed to be the main constraint in the operation of the network.

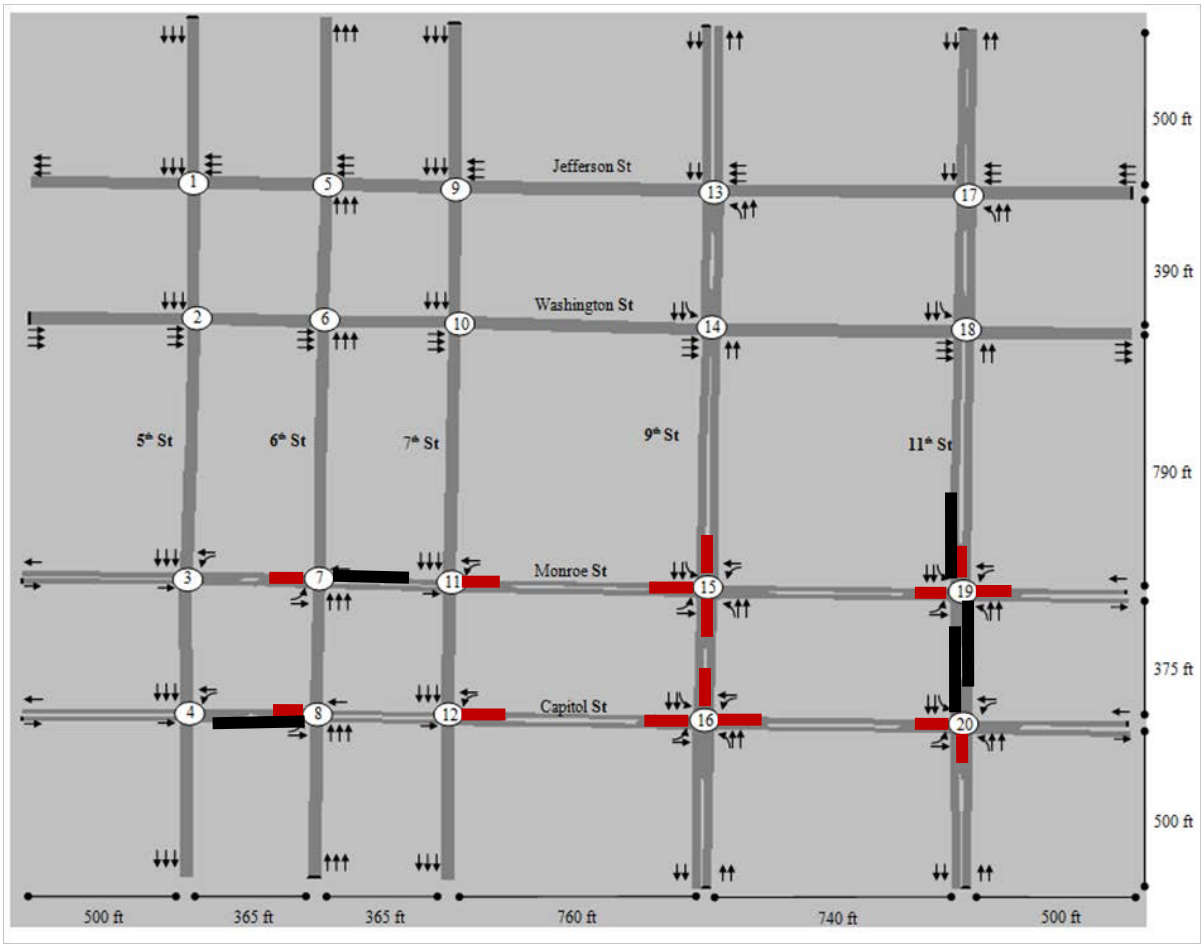


Figure 4.28. Location of queue backups for ADP and Q-learning in oversaturated 4x5 network

4.6 4x5 Network, Oversaturated Conditions - Uneven Demands

In this case, the same network used in the previous scenario is tested with reduced demands in one of the directions of traffic. The objective of this experiment was to determine if under less demanding loads, the max-plus algorithm is also able to improve the performance of a RL implementation. It is expected to have increased benefits when using a strategy to group agents in scenarios with heavier demands in one direction of traffic. Two implementations were compared: one using the Q-learning implementation from the previous case, and one that is analogous but also has the max-plus algorithm.

The demands were reduced in the E-W direction to about one third of their demands in the previous case, thus the heavier traffic will be in the N-S direction and will carry very high volumes that may also result in blockages if not managed properly.

The resulting network throughput for the two implementations is shown in Figure 4.29. The max-plus algorithm resulted in a similar learning curve than in the case without it. However, the early learning stages favored max-plus showing faster discovery of useful solutions to increase throughput. This can be observed in the first 100 replications, where the moving average is slightly higher for the max-plus implementation.

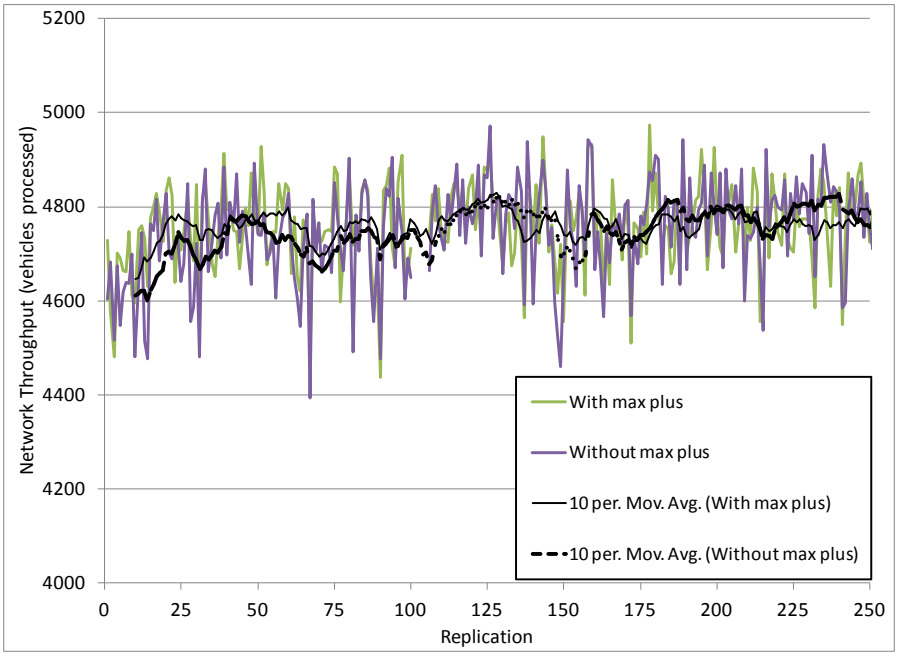


Figure 4.29. Learning curve for network throughput with and without Max-plus algorithm in the oversaturated 4x5 network with uneven demands

A closer look at the benefits of max-plus can be found in a similar format to that shown for the previous case with even demands. This is shown in Figure 4.30, where the average number of stops is plotted versus the total network throughput. The max-plus algorithm resulted in fewer average stops per vehicle and a higher number of vehicle processed by

the network. The number of stops was reduced by 5%, or 0.13 fewer stops per vehicles, for a total of more than 600 fewer stops in the network. Similarly, the average throughput was increased with max-plus by 34 vehicles, which corresponds to a 1% increase. It is also noted that the effects with uneven demands seemed to be in similar proportion to those observed in the case with even demands.

Thus, with uneven demands, coordination due to an external algorithm coupled to the learning strategies also resulted in benefits for the network as a whole. However, the current max-plus implementation may result in competing coordination between adjacent intersections, thus indicating that there is potential for improved implementations where a given coordinating direction should be emphasized over an extended area without overloading the links. This may result in significant network-wide improvements as the coordination directions will be explicitly decided over corridors instead of immediate neighbors.

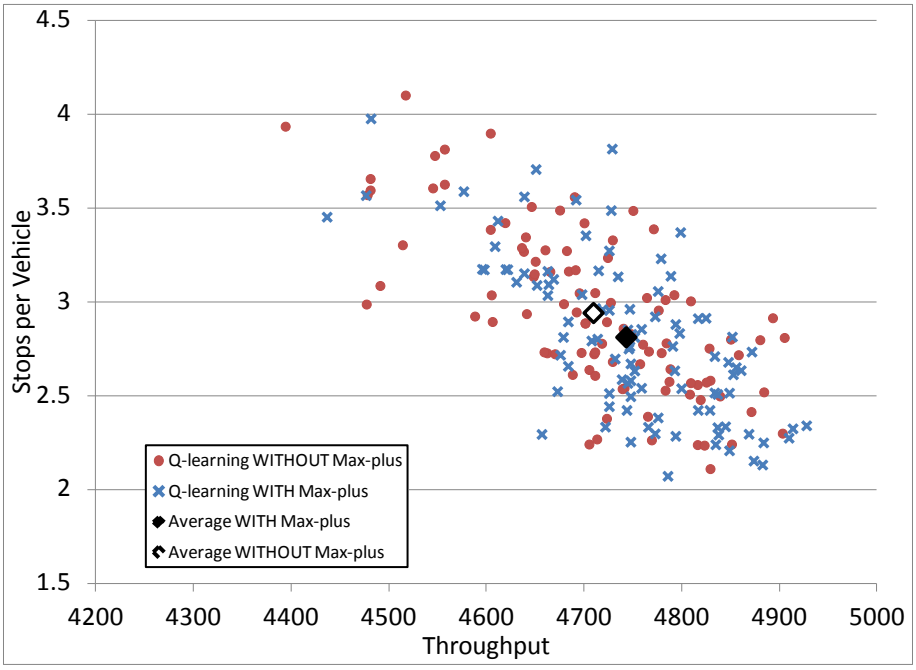


Figure 4.30. Effects of Max-plus algorithm on the Q-learning implementation in the oversaturated 4x5 network – uneven demands

CHAPTER 5. CONCLUSIONS AND FUTURE WORK

This study explores the utilization of reinforcement learning (RL) agents for traffic signal control in a variety of scenarios, with emphasis on oversaturated conditions. The algorithms of choice for this study were Q-learning and an approximate dynamic programming (ADP) with a post-decision state variable. These strategies were implemented using a commercially available microscopic traffic simulator (VISSIM) and its communication interface, which allowed for the manipulation of the traffic signals in real time. In addition, an explicit coordinating mechanism (the max-plus algorithm) was included in one of the RL algorithms to determine its benefits with high traffic demands.

A series of scenarios were created to test the RL agents. Their complexity increased from an oversaturated isolated intersection, to an arterial in undersaturated conditions, to a 2x5 network in both undersaturation and oversaturation, and finally to a 4x5 network in oversaturation with even and uneven directional demands.

Results showed that agents with RL algorithms (ADP and Q-learning) were able to manage the traffic signals efficiently in both undersaturation and oversaturation. This was observed in all the cases analyzed in this study. In the isolated intersection, the signals processed vehicles at short discharge headways and provided green times in a similar proportion to the actual demand for left-turns and through movements. Through phases were displayed more often, reducing lost times in frequent transitions to left-turning movements that had lower demands. Also, improved performance was found if the state of the system not only considered the number of vehicles in the links, but also an estimate of the time vehicles have spent in the link.

For the arterial in undersaturation, the agents continuously provided green to approaches with demand at intersections with no opposing traffic and also favored coordination for the two adjacent intersections with conflicting volumes. Coordination was emphasized in the direction of heavier traffic, as expected, and performance was

similar to that provided by signals optimized by TRANSYT7F. Implementations that included features such as incentives for providing green to oncoming vehicles from neighboring intersections showed benefits over those that did not.

In a 2x5 network in undersaturation, the RL agents prevented queue spillbacks for through vehicles, but left-turn pockets were momentarily blocked due to the permitted nature of the turning movements. The total number of vehicles processed fluctuated around the total expected demand for this scenario, indicating no increase in residual queues at the end of the study period. In oversaturation, the agents were tested with and without communication capabilities to illustrate the need to provide information on adjacent intersections in order to prevent queue spillbacks. Results clearly showed that the performance of the network was improved with communication capabilities, in this case by informing of potential downstream blockages.

Lastly, the RL agents were tested in a realistic 4x5 network in oversaturation with even and uneven directional demands. In the first case, scenarios with ADP and Q-learning were implemented separately, in addition to a scenario using a Q-learning strategy with an explicit coordination strategy using the max-plus algorithm. The performance of the three implementations was similar, with improvements for the case with the max-plus algorithm. These results were comparable to those obtained by optimizing the signals with TRANSYT7F. Analysis of the queues in the network showed that most problematic areas occurred at intersections with only one through lane, especially where left-turn lanes had long queues and blocked the through movement. An additional scenario with heavier demands in one direction of traffic (N-S) was created to determine if the max-plus algorithm could offer additional benefits to the network. Results showed a trend to obtain increased throughput and reduced number of stops when the outcome of the max-plus algorithm was added to the reward structure such that the coordinated direction of traffic was emphasized. This indicates that there is potential benefits using explicit mechanisms to coordinate agents, and opens the discussion for additional exploration of such mechanisms and how to incorporate them into the RL process.

In summary, results presented in this study shows that reinforcement learning agents can efficiently control the traffic signals in realistic networks and oversaturated conditions when implemented with communication capabilities designed to prevent queue spillbacks and promote signal coordination. Oversaturation is especially challenging for the agents to manage queues, but results indicate that even in this conditions the traffic signals prevented spillbacks and gridlocks and at a level comparable to state-of-practice traffic optimization software.

Future work includes further experimentation to expand the use of agents in larger networks and varying traffic demands. Special attention should also be given to alternate algorithms or alternate implementations of explicit coordinating strategies in order to increase the efficiency of the network, including other implementations of the max-plus algorithm and its coupling to ADP and Q-learning strategies. Additional restrictions to the max-plus algorithm to limit multidirectional coordination between adjacent intersections may result in significant improvements and will be pursued in future applications.

The implementation of these strategies to undersaturated networks (including a coordinating mechanism) could also result in benefits compared to current state-of-practice signal timing, mostly due to their flexibility to face unexpected changes in demands. Furthermore, scenarios where a network transitions between undersaturated and oversaturated conditions could also be improved using reinforcement learning.

Lastly, a number of questions remain open in terms of further enhancements in the performance with increased communication capabilities, including not only information passing (state and reward sharing), but also advice exchange and negotiating strategies suitable for real-time applications.

CHAPTER 6. REFERENCES

Abdulhai, B., Pringle, R., and Karakoulas, G.J. (2003) Reinforcement Learning for True Adaptive Traffic Signal Control, ASCE Journal of Transportation Engineering. Vol. 129, No 3, pp. 278-285.

Appl, M., and Brauer, W. (2000) Fuzzy Model-Based Reinforcement Learning. Presented at the European Symposium on Intelligent Techniques, Germany.

Bakker, B., Steingrover, M., Schouten, R., Nijhuis, and E., Kester, L. (2005) Cooperative Multi-agent Reinforcement Learning of Traffic Lights. In: Gama, J., Camacho, R.,Brazdil, P.B., Jorge, A.M., Torgo, L. (eds.) ECML 2005. LNCS (LNAI), vol. 3720. Springer, Heidelberg.

Barto, A.G., Sutton, R.S., and Anderson, C.W. (1983) Neuronlike adaptive elements that can solve difficult learning control problems. IEEE Transactions on Systems, Man, and Cybernetics, 13:835-846.

Bellman, R. (1957) Dynamic Programming, Princeton University Press, Princeton.

Bellman, R., and Dreyfus, S. (1959) Functional Approximations and Dynamic Programming, Mathematical Tables and Other Aids to Computation 13, 247–251.

Bertsekas, D., and Tsitsiklis, J. (1996) Neuro-Dynamic Programming, Athena Scientific, Belmont, MA.

Bingham, E. (1998) Neurofuzzy Traffic Signal Control. Master's thesis, Dept. of Engineering Physics and Mathematics, Helsinki Univ. of Technology, Helsinki, Finland.

Bingham, E. (2001) Reinforcement Learning in Neurofuzzy Traffic Signal Control, European Journal of Operations Research. Vol. 131, No. 2, pp. 232–241.

Cai, C., Kwong Wong, C., and Heydecker, B.G. (2009) Adaptive Traffic Signal Control Using Approximate Dynamic Programming, Transportation Research Part C: Emerging Technologies, vol. 17, pp. 456-474.

Camponogara, E., and Kraus, W. Jr. (2003) Distributed Learning Agents in Urban Traffic Control. In: Progress in Artificial Intelligence: Proceedings of the 11th Portuguese Conference on Artificial Intelligence (EPIA).

Choy, M. C., Cheu, R. L., Srinivasan, D., and Logi, F. (2003) Real-time Coordinated Signal Control Using Agents with Online Reinforcement Learning. In Proceedings of the 82nd Transportation Research Board Annual Meeting. Washington, D.C.

Crick, C., and Pfeffer, A. (2003) Loopy belief propagation as a basis for communication in sensor networks. In Proceedings of Uncertainty in Artificial Intelligence (UAI).

Dearden, R., Friedman, N., and Russell, S. (1998) Bayesian Q-Learning. Fifteenth National Conference on Artificial Intelligence (AAAI), Madison, WI.

De Oliveira, D., and Bazzan, A. L.C. (2006a) Emergence of Traffic Lights Synchronization. Proceedings 20th European Conference on Modeling and Simulation. Germany.

De Oliveira, D., Bazzan, A. L.C., Castro da Silva, B., Basso, E.W., and Nunez, L. (2006b) Reinforcement Learning based Control of Traffic Lights in Non-stationary Environments: A Case Study in a Microscopic Simulator. Fourth European Workshop on Multi-Agent Systems. Portugal.

Gartner, N.H. (1983) OPAC: a demand-responsive strategy for traffic signal control. Transportation Research Record 906, 75–81.

Gosavi, A. (2009) Reinforcement Learning: A tutorial Survey and Recent Advances. INFORMS Journal on Computing, Vol. 21, No. 2, pp. 178-192.

Hajbabaie, A., Medina, J.C., Benekohal, R.F. (2011) Traffic Signal Coordination and Queue Management in Oversaturated Intersections. NEXTRANS Project No. 047IY02, 2011.

Humphrys, M. (1995) W-learning: Competition Among Selfish Q-learners. Technical Report no.362, University of Cambridge, UK.

Humphrys, M. (1997) Action selection Methods using Reinforcement Learning. PhD Thesis. University of Cambridge, UK.

Junges, R., and Bazzan, A.L.C. (2007) Modelling Synchronization of Traffic Lights as a DCOP. Proceedings of the 5th European Workshop on Multiagent Systems, pp.564-579, Tunisia.

Kok, J. R., Vlassis, N. (2005) Using the max-plus algorithm for multiagent decision making in coordination graphs. In RoboCup-2005: Robot Soccer World Cup IX, Osaka, Japan.

Kok, J. R., Vlassis N. (2006) Collaborative multiagent reinforcement learning by payoff propagation. J Machine Learn Res. 7: 1789-1828.

Kuyer L., Whiteson, S., Bakker, B., and Vlassis, N. (2008) Multiagent Reinforcement Learning for Urban Traffic Control using Coordination Graphs. In Proc. 19th European Conference on Machine Learning, Antwerp, Belgium.

Medina, J.C., Hajbabaie, A., Benekohal, R.F. (2010) Arterial Traffic Control Using Reinforcement Learning Agents and Information from Adjacent Intersections in the State and Reward Structure. Presented at the 13th International IEEE Annual Conference on Intelligent Transportation Systems. Madeira, Portugal.

Medina, J.C., Benekohal, R.F. (2011) Reinforcement Learning Agents for Traffic Signal Control in Oversaturated Networks. Presented at the First TD&I Conference of the ASCE, Chicago, IL.

Murphy, K., Weiss, K., and Jordan, M. (1999) Loopy belief propagation for approximate inference: An empirical study. In Proceedings of Uncertainty in Artificial Intelligence (UAI), Stockholm, Sweden.

Oliveira, D., Bazzan, A. L. C., and Lesser, V. (2005) Using Cooperative Mediation to Coordinate Traffic Lights: a Case Study. In: Fourth International Joint Conference on Autonomous Agents and Multiagent Systems, Utrecht. New York : ACM, v. 1.

Oliveira, D., Ferreira JR., P. R., Bazzan, A. L. C., and Kluegl, F. (2004) A Swarm-based Approach for Selection of Signal Plans in Urban Scenarios. In: IV International Workshop on Ant Colony Optimization and Swarm Intelligence (ANTS), Brussels.

Pendrith, M. (1994) On Reinforcement Learning of Control Actions in Noisy Non-Markovian Domains. UNSW-CSE-TR-9410. University of South Wales, Australia.

Peng, J. and Williams, R. (1991) Incremental Multi-step Q-learning. Machine Learning 22:282-290.

Peng, J. (1993) Efficient Dynamic Programming-Based Learning for Control. PhD Dissertation, Northeastern University, Boston.

Powell, W.B. (2007) Approximate Dynamic Programming: Solving the Curses of Dimensionality, Wiley, New York.

Powell, W.B. (2010). Approximate Dynamic Programming – II: Algorithms. In: Wiley Encyclopedia of Operations Research and Management Science. John Wiley & Sons, Inc.

Richter, S., Aberdeen, D., and Yu, J. (2007) Natural Actor-Critic for Road Traffic Optimisation. In: Advances in Neural Information Processing Systems MIT Press , Cambridge, MA , pp. 1169-1176.

Robert L. Gordon, P.E., Warren Tighe, P.E. (2005) Traffic control systems handbook. Federal Highway Administration. Publication Number FHWA-HOP-06-006.

Robertson, D.I., Bertherton, R.D. (1974) Optimum control of an intersection for any known sequence of vehicular arrivals. In: Proceedings of the second IFACIFIP-IFORS Symposium on Traffic Control and Transportation system, Monte Carlo.

Robbins, H., and Monro, S. (1951) A Stochastic Approximation Method, Annals of Mathematical Statistics, Vol. 22, pp.400–407.

Rummery, G.A., Niranjan, M. (1994) On-line Q-learning using Connectionist Systems. CUED/F-INFENG/TR 166, Cambridge University, UK.

Sutton, R.S. (1988) Learning to Predict by Methods of Temporal Difference. Machine Learning, 3:9-44.

Sutton, R.S., and Barto, A.G. (1998) Reinforcement Learning: An Introduction. MIT Press.

Teodorvic D., V. Varadarajan, J. Popovic, M. R. Chinnaswamy, S. Ramaraj. (2006) Dynamic programming - neural network real-time traffic adaptive signal control algorithm. Annals of Operation Research, pp-123-131.

Tesauro, G. (1992) Practical Issues in Temporal Difference Learning. Advances in Neural Information Processing Systems 4, pp.259-266, San Mateo, CA, Morgan Kaufmann.

Thorpe, T. (1997) Vehicle Traffic Light Control Using SARSA, Masters Thesis, Department of Computer Science, Colorado State University.

N, Vlassis, R. Elhorst, and J. R. Kok. (2004) Anytime algorithms for multiagent decision making using coordination graphs. In Proceedings of the International Conference on Systems, Man, and Cybernetics (SMC), The Hague, The Netherlands.

Wainwright, M. J., Jaakkola, T. S., and Willsky, A. S. (2004) Tree consistency and bounds on the performance of the max-product algorithm and its generalizations. Statistics and Computing, 14:143–166.

Watkins, C.J.C.H. (1989) Learning from delayed rewards. PhD Thesis, King's College, Cambridge, England.

Watkins, C.J.C.H. and Dayan, P. (1992) Technical note: Q-learning. Machine Learning 8. Pp. 279-292.

Werbos, P. J. (1987) Building and understanding adaptive systems: A statistical/numerical approach to factory automation and brain research. IEEE Transactions on Systems, Man., and Cybernetics, 17:7-20.

Wiering, M. and Schmidhuber, J. (1997) HQ-Learning. Adaptive Behavior, 6 (2), 219-246.

Wiering, M. and Schmidhuber, J. (1998) Fast Online Q(λ). Machine Learning, 33 (1), 105-115.

Wiering, M. (2000) Multi-Agent Reinforcement Learning for Traffic Light Control. In: Proc. 17th International Conf. on Machine Learning, pp. 1151–1158.

Xie, Y. (2007) Development and Evaluation of an Arterial Adaptive Traffic Signal Control System using Reinforcement Learning, Doctoral Dissertation, Texas A&M University: College Station, TX.

Yedidia, J. S., Freeman, W. T., and Weiss, Y. (2003) Understanding belief propagation and its generalizations. In Exploring Artificial Intelligence in the New Millennium, chapter 8, pages 239–269. Morgan Kaufmann Publishers Inc.

Zhang, Y., Xie, Y., and Ye, Y. (2007) Development and Evaluation of a Multi-Agent Based Neuro-Fuzzy Arterial Traffic Signal Control System, Texas A&M University, Report Number SWUTC/07/473700-00092-1, 2007.