Multiagent Reinforcement Learning for Urban Traffic Control using Coordination Graphs

Lior Kuyer[1], Shimon Whiteson[1], Bram Bakker[1], and Nikos Vlassis[2]

[1] Informatics Institute, University of Amsterdam
Kruislaan 403, 1098 SJ Amsterdam, The Netherlands
{lkuijer,whiteson,bram}@science.uva.nl
[2] Department of Production Engineering and Management
Technical University of Crete, Chania, Greece
vlassis@dpem.tuc.gr

Abstract. Since traffic jams are ubiquitous in the modern world, optimizing the behavior of traffic lights for efficient traffic flow is a critically important goal. Though most current traffic lights use simple heuristic protocols, more efficient controllers can be discovered automatically via multiagent reinforcement learning, where each agent controls a single traffic light. However, in previous work on this approach, agents select only locally optimal actions without coordinating their behavior. This paper extends this approach to include explicit coordination between neighboring traffic lights. Coordination is achieved using the max-plus algorithm, which estimates the optimal joint action by sending locally optimized messages among connected agents. This paper presents the first application of max-plus to a large-scale problem and thus verifies its efficacy in realistic settings. It also provides empirical evidence that max-plus performs well on cyclic graphs, though it has been proven to converge only for tree-structured graphs. Furthermore, it provides a new understanding of the properties a traffic network must have for such coordination to be beneficial and shows that max-plus outperforms previous methods on networks that possess those properties.

Key words: multiagent systems, reinforcement learning, coordination graphs, max-plus, traffic control

1 Introduction

Traffic jams are ubiquitous in the modern world and are getting worse, due to rapidly increasing populations and vehicle usage rates. They commonly occur in urban settings, where traffic lights are the most typical control mechanism. Existing road infrastructure is often strained to its limits and expansion is infeasible due to spatial, environmental, and economic constraints. Therefore, optimizing the behavior of traffic lights for efficient traffic flow is a critically important goal.

In practice, most traffic lights use very simple protocols that merely alternate red and green lights for fixed intervals. The interval lengths may change during peak hours but are not otherwise optimized. Since such controllers are far

from optimal, several researchers have investigated the application of machine learning to automatically discover more efficient controllers. The methods employed include fuzzy logic [5], neural networks [12] and evolutionary algorithms [7]. These methods perform well but can only handle networks with a relatively small number of controllers.

Since traffic control is fundamentally a problem of sequential decision making, it is perhaps best suited to the framework of reinforcement learning, in which an agent learns from trial and error via direct interaction with its environment. Each action results in immediate rewards and new observations about the state of the world. Over time, the agent learns a control policy that maximizes the expected long-term reward it receives.

One way to apply reinforcement learning to traffic control is to train a single agent to control the entire system, i.e. to determine how every traffic light in the network is set at each timestep. However, such centralized controllers scale very poorly, since the size of the agent's action set is exponential in the number of traffic lights.

An alternative approach is to view the problem as a multiagent system where each agent controls a single traffic light [3, 14]. Since each agent observes only its local environment and selects only among actions related to one traffic light, this approach can scale to large numbers of agents. The primary limitation is that the individual agents do not coordinate their behavior. Consequently, agents may select individual actions that are locally optimal but that together result in global inefficiencies.

This paper extends the reinforcement learning approach to traffic control by using cooperative learning and explicit coordination among agents. We make the relaxing assumption that an agent is affected only by those agents with a direct influence on its environment, i.e. its neighbors in the network. Under this assumption, the global coordination problem may be decomposed into a set of local coordination problems and can be solved with the use of *coordination graphs* [9].

Since the system must perform under time constraints, an efficient method for finding optimal joint actions in such graphs is required. For this reason we apply *max-plus* [10], which estimates the optimal joint action by sending locally optimized messages among connected agents. It also allows the agents to report their current best action at any time (even if the action found so far may be suboptimal).

This paper makes several contributions. First, it augments the existing framework of reinforcement learning for traffic control by allowing scalable coordination of neighboring traffic lights. Second, it presents the first application of max-plus to a large-scale problem and thus verifies its efficacy in realistic settings. Third, it provides empirical evidence that max-plus performs well on cyclic graphs, though it has been proven to converge only for tree-structured graphs. Fourth, it provides a new understanding of the properties a traffic network must have for such coordination to be beneficial and shows that max-plus outperforms previous methods on networks that possess those properties.

The remainder of this paper is organized as follows. Section 2 introduces the traffic model used in our experiments. Section 3 describes the traffic control problem as a reinforcement learning task. Section 4 describes coordination graphs and the max-plus algorithm and Section 5 describes how max-plus is applied to the traffic control problem. Section 6 presents experimental results and Section 7 discusses these results. Section 8 outlines directions for future work and Section 9 concludes.

2 Traffic Model

All experiments presented in this paper were conducted using The Green Light District (GLD) traffic simulator[1] [3, 14]. GLD is a *microscopic* traffic model, i.e. it simulates each vehicle individually, instead of simply modeling aggregate properties of traffic flow. The dynamic variables of the model represent microscopic properties such as the position and velocity of each vehicle. Vehicles move through the network according to their physical characteristics (e.g. length, speed, etc.), fundamental rules of motion, and predefined rules of driver behavior. GLD's simulation is based on *cellular automata*, in which discrete, partially connected cells can occupy various states. For example, a road cell can be occupied by a vehicle or be empty. Local transition rules determine the dynamics of the system and even simple rules can lead to a highly dynamic system.

The GLD infrastructure consists of roads and nodes. A road connects two nodes, and can have several lanes in each direction. The length of each road is expressed in cells. A node is either an intersection where traffic lights are operational or an edge node. There are two types of agents that occupy such an infrastructure: vehicles and traffic lights (or intersections). All agents act autonomously and are updated every timestep. Vehicles enter the network at edge nodes and each edge node has a certain probability of generating a vehicle at each timestep. Each generated vehicle is assigned one of the other edge nodes as a destination. The distribution of destinations for each edge node can be adjusted.

There are several types of vehicles, defined by their speed, length, and number of passengers. In this paper, all vehicles have equal length and an equal number of passengers. The state of each vehicle is updated every timestep. It either moves the distance determined by its speed and the state around it (e.g. vehicles in front may limit how far it can travel) or remains in the same position (e.g. the next position is occupied or a traffic light prevents its lane from moving).

When a vehicle crosses an intersection, its driving policy determines which lane it goes to next. Once a lane is selected, the vehicle cannot switch to a different lane. For each intersection, there are several light configurations that are safe. At each timestep, the intersection must choose from among these configurations, given the current state.

Figure 1 shows an example GLD intersection. It has four roads, each consisting of four lanes (two in each direction). Vehicles occupy n cells of a lane,

[1] available at http://sourceforge.net/projects/stoplicht

depending on their length. Traffic on a given lane can only travel in the directions allowed on that lane. This determines the possible safe light configurations. For example, the figure shows a lane where traffic is only allowed to travel straight or right.

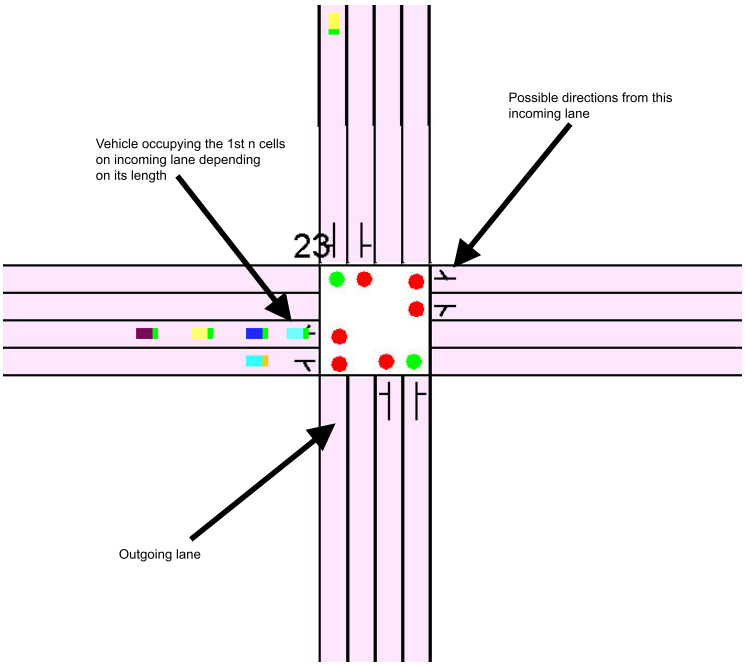

Fig. 1. An example GLD intersection.

The behavior of each vehicle depends on how it selects a path to its destination node and how it adjusts its speed over time. In our experiments, the vehicles always select the shortest path to their destination node. In previous work [3, 14], vehicles always traveled at constant speed and only a single vehicle could cross an intersection at each timestep. We extend the simulator to allow more dynamic behavior. Three speeds (2, 4, or 6 cells per timestep) are now possible. Vehicles enter the network with a speed of 4 and at each timestep there is a 78% chance the vehicle will keep its current speed when it is 4 and an 88% chance when it is either 2 or 6. Furthermore, multiple vehicles from a single lane can now cross an intersection during each timestep. The number depends on the vehicles' speed and on the state of the destination lanes.

3 Reinforcement Learning for Urban Traffic Control

Several techniques for learning traffic controllers with model-free reinforcement learning methods like Sarsa [13] or Q-learning [1, 11] have previously been developed. However, they all suffer from the same problem: they do not scale to

large networks since the size of the state space grows rapidly. Hence, they are either applied only to small networks or are used to train homogeneous controllers (by training on a single isolated intersection and copying the result to each intersection in the network).

A more tractable approach is to use model-based reinforcement learning, in which the transition and reward functions are estimated from experience and then used to find a policy via planning methods like *dynamic programming* [4]. A full transition function would have to map the location of every vehicle in the system at one timestep to the location of every vehicle at the next timestep. Doing so is clearly infeasible, but learning a model is nonetheless possible if a *vehicle-based* representation [14] is used. In this approach, the global state is decomposed into local states based on each individual vehicle. The transition function maps one vehicle's location at a given timestep to its location at the next timestep. As a result, the number of states grows linearly in the number of cells and can scale to much larger networks. Furthermore, the transition function can generalize from experience gathered in different locations, rather than having to learn separate mappings for each location.

To represent the model, we need only keep track of the number of times each transition (s, a, s') has occurred and each state-action pair (s, a) has been reached. The transition model can then be estimated via the maximum likelihood probability $\frac{|(s,a,s')|}{|(s,a)|}$. Hence, each timestep produces new data which is used to update the model. Every time the model changes, the value function computed via dynamic programming must be updated too. However, rather than having to update each state, we can update only the states most likely to be affected by the new data, using an approach based on *prioritized sweeping* [2]. The remainder of this section describes the process of learning the model in more detail.

Given a vehicle-based representation, the traffic control problem consists of the following components:

- $s \in S$: the fully observable global state
- $i \in I$: an intersection controller
- $a \in A$: an action, which consists of setting to green a subset of the traffic lights at the intersection; $A_i \subseteq A$ is the subset of actions that are safe at intersection i
- $l \in L$: a traffic lane; $L_i \subseteq L$ is the subset of incoming lanes for intersection i
- $p \in P$: a position; $P_l \subseteq P$ is the subset of positions for lane l

The global transition model is $P(s'|s, a)$ and the global state s decomposes into a vector of local states, $s = \langle s_{p_{l_i}} \rangle$, with one for each position in the network. The action-value function decomposes as:

$$Q(s, a) = \sum_i Q_i(s_i, a_i) \tag{1}$$

where

$$Q_i(s_i, a_i) = \sum_{l_i} \sum_{p_{l_i}} Q_{p_{l_i}}(s_{p_{l_i}}, a_i) \tag{2}$$

The vehicle-based update rule is then given by:

$$Q_{p_{l_i}}(s_{p_{l_i}}, a_i) := \sum_{s'_{p_{l_i}} \in S'} P(s'_{p_{l_i}}|a_i, s_{p_{l_i}})[r(s_{p_{l_i}}, s'_{p_{l_i}}) + \gamma V(s'_{p_{l_i}})] \tag{3}$$

where S' are all possible states that can be reached from $s_{p_{l_i}}$ given the current traffic situation and the vehicle's properties (e.g. its speed and length). Like Wiering, we use a learning rate $\gamma = 0.9$. $V(s_{p_{l_i}})$ estimates the expected waiting time at p_{l_i} and is given by:

$$V(s_{p_{l_i}}) := \sum_{a_i} P(a_i|s_{p_{l_i}}) Q(s_{p_{l_i}}, a_i) \tag{4}$$

The transition model can be estimated using maximum likelihoods by counting state transitions and corresponding actions. The update is given by:

$$P(s'_{p_{l_i}}|s_{p_{l_i}}, a_i) := \frac{C(s_{p_{l_i}}, a_i, s'_{p_{l_{ij}}})}{C(s_{p_{l_i}}, a_i)} \tag{5}$$

where $C(\cdot)$ is a function that counts the number of times the event occurs. To estimate $V(s_{p_{l_i}})$, we also need to estimate the probability that a certain action will be taken given the state, which is done using the following update:

$$P(a_i|s_{p_{l_i}}) := \frac{C(s_{p_{l_i}}, a_i)}{C(s_{p_{l_i}})} \tag{6}$$

The global reward function decomposes as:

$$r(s, s') = \sum_i \sum_{l_i} \sum_{p_{l_i}} r(s_{p_{l_i}}, s'_{p_{l_i}}) \tag{7}$$

and

$$r(s_{p_{l_i}}, s'_{p_{l_i}}) = \begin{cases} 0 & s_{p_{l_i}} \neq s'_{p_{l_i}} \\ -1 \text{ otherwise} \end{cases} \tag{8}$$

Given the current model, the optimal value function is estimated using dynamic programming with a fixed number of iterations. Wiering [14] performs only one iteration per timestep and uses ϵ-greedy exploration to ensure the estimated model obtains sufficiently diverse data.

Bakker et al. [3] extend Wiering's approach by including congestion information in the state representation. The value function $Q_{p_{l_i}}(s_{p_{l_i}}, a_i)$ is extended to $Q_{p_{l_i}}(s_{p_{l_i}}, c_{dest}, a_i)$ where $c_{dest} \in \{0, 1\}$ is a single bit indicating the congestion level at the next lane for the vehicle currently at p_{l_i}. If the congestion at the next lane exceeds some threshold then $c_{dest} = 1$ and otherwise it is set to 0. This extension allows the agents to learn different state transition probabilities and value functions when the outbound lanes are congested. This method has been shown to outperform Wiering's approach on a saturated network. The cost of including such congestion information is a larger state space and potentially slower learning. It also requires the vehicles to communicate with the controllers, since the latter need to know the destination lanes of each vehicle.

4 Coordination Graphs

The primary limitation of the approaches developed by Wiering and Bakker et al. is that the individual agents do not coordinate their behavior. Consequently, agents may select individual actions that are locally optimal but that together result in global inefficiencies. Coordinating actions can be difficult since the size of the joint action space is exponential in the number of agents. However, in many cases, the best action for a given agent may depend on only a small subset of the other agents. If so, the global reward function can be decomposed into local functions involving only subsets of agents. The optimal joint action can then be estimated by finding the joint action that maximizes the sum of the local rewards.

A *coordination graph* [9], which can be used to describe the dependencies between agents, is an undirected graph $G = (V, E)$ in which each node $i \in V$ represents an agent and each edge $e(i, j) \in E$ between agents i and j indicates a dependency between them. The global coordination problem is then decomposed into a set of local coordination problems, each involving a subset of the agents. Since any arbitrary graph can be converted to one with only pairwise dependencies [16], the global action-value function can be decomposed into pairwise value functions given by:

$$Q(s, a) = \sum_{i,j \in E} Q_{ij}(s, a_i, a_j) \tag{9}$$

where a_i and a_j are the corresponding actions of agents i and j, respectively. Using such a decomposition, the *variable elimination* [9] algorithm can compute the optimal joint action by iteratively eliminating agents and creating new conditional functions that compute the maximal value the agent can achieve given the actions of the other agents on which it depends. Although this algorithm always finds the optimal joint action, it is computationally expensive, as the execution time is exponential in the induced width of the graph [15]. Furthermore, the actions are known only when the entire computation completes, which can be a problem for systems that must perform under time constraints. In such cases, it is desirable to have an *anytime* algorithm that improves its solution gradually.

One such algorithm is *max-plus* [8, 10], which approximates the optimal joint action by iteratively sending locally optimized messages between connected nodes in the graph. While in state s, a message from agent i to neighboring agent j describes a local reward function for agent j and is defined by:

$$\mu_{ij}(a_j) = \max_{a_i} \{Q_{ij}(s, a_i, a_j) + \sum_{k \in \Gamma(i) \backslash j} \mu_{ki}(a_i)\} + c_{ij} \tag{10}$$

where $\Gamma(i) \backslash j$ denotes all neighbors of i except for j and c_{ij} is either zero or can be used to normalize the messages. The message approximates the maximum value agent i can achieve for each action of agent j based on the function defined between them and incoming messages to agent i from other connected agents

(except j). Once the algorithm converges or time runs out, each agent i can select the action

$$a_i^* = \arg\max_{a_i} \sum_{j \in \Gamma(i)} \mu_{ji}(a_i) \tag{11}$$

Max-plus has been proven to converge to the optimal action in finite iterations, but only for tree-structured graphs, not those with cycles. Nevertheless, the algorithm has been successfully applied to such graphs [6, 10, 16].

5 Max-Plus for Urban Traffic Control

Max-plus enables agents to coordinate their actions and learn cooperatively. Doing so can increase robustness, as the system can become unstable and inconsistent when agents do not coordinate. By exploiting coordination graphs, max-plus minimizes the expense of computing joint actions and allows them to be approximated within time constraints.

In this paper, we combine max-plus with Wiering's model-based approach to traffic control. We use the vehicle-based representation defined in Section 3 but add dependence relationships between certain agents. If $i, j \in J$ are two intersections connected by a road, then they become neighbors in the coordination graph, i.e. $i \in \Gamma(j)$ and $j \in \Gamma(i)$. The local value functions are:

$$Q_i(s_i, a_i, a_j) = \sum_{l_i} \sum_{p_{l_i}} Q_{p_{l_i}}(s_{p_{l_i}}, a_i, a_j) \tag{12}$$

Using the above, we can define the pairwise value functions used by max-plus:

$$Q_{ij}(s, a_i, a_j) = \sum_{p_{l_i}} O_{p_{l_i}} Q_{p_{l_i}}(s_{p_{l_i}}, a_i, a_j) + \sum_{p_{l_j}} O_{p_{l_j}} Q_{p_{l_j}}(s_{p_{l_j}}, a_j, a_i) \tag{13}$$

where $O_{p_{l_i}}$ is a binary operator which indicates occupancy at p_{l_i}:

$$O_{p_{l_i}} = \begin{cases} 0 \; p_{l_i} \text{ not occupied} \\ 1 \text{ otherwise} \end{cases} \tag{14}$$

These local functions are plugged directly into Equation 10 to implement max-plus. Note that the functions are symmetric such that $Q_{ij}(s, a_i, a_j) = Q_{ji}(s, a_j, a_i)$. Thus, using Equation 13, the joint action can be estimated directly by the max-plus algorithm. Like Wiering, we use one iteration of dynamic programming per timestep and ϵ-greedy exploration. We also limit max-plus to 3 iterations per timestep.

Note that there are two levels of value propagation among agents. On the lower level, the vehicle-based representation enables estimated values to be propagated between neighboring agents and eventually through the entire network, as in Wiering's approach. On the higher level, agents use max-plus when computing joint actions to inform their neighbors of the best value they can achieve, given the current state and the values received from other agents.

Using this approach, agents can learn cooperative behavior, since they share value functions with their neighbors. Furthermore, they can do so efficiently, since the number of value functions is linear in the induced width of the graph. Stronger dependence relationships could also be modeled, i.e. between intersections not directly connected by a road, but we make the simplifying assumption that it is sufficient to model the dependencies between immediate neighbors in the traffic network.

6 Results

In this section, we compare the novel approach described in Section 5 to the TC-1 (Traffic Controller 1) developed by Wiering [14] and the TC-SBC (Traffic Controller with State Bit for Congestion) extension of Bakker et al. [3]. Wiering compared TC-1 to two heuristic strategies, one that always sets the lights at each intersection to maximize throughput and another that always gives right-of-way to the longest queue. These heuristics perform well in light traffic but TC-1 substantially outperforms them in heavy traffic. Therefore, we focus our experiments on comparisons between the novel method, TC-1, and TC-SBC in highly saturated conditions.

These experiments are designed to test the hypothesis that, under highly saturated conditions, coordination is beneficial when the amount of *local traffic* is small. Local traffic consists of vehicles that cross a single intersection and then exit the network, thereby interacting with just one learning agent. If this hypothesis is correct, coordinated learning with max-plus should substantially outperform TC-1 and TC-SBC when most vehicles pass through multiple intersections.

In particular, we consider three different scenarios. In the *baseline* scenario, the traffic network includes routes, i.e. paths from one edge node to another, that cross only a single intersection. Since each vehicle's destination is chosen from a uniform distribution, there is a substantial amount of local traffic. In the *nonuniform destinations* scenario, the same network is used but destinations are selected to ensure that each vehicle crosses two or more intersections, thereby eliminating local traffic. To ensure that any performance differences we observe are due to the absence of local traffic and not just to a lack of uniform destinations, we also consider the *long routes* scenario. In this case, destinations are selected uniformly but the network is altered such that all routes contain at least two intersections, again eliminating local traffic.

While a small amount of local traffic will occur in real-world scenarios, the vast majority is likely to be non-local. Thus, the baseline scenario is used, not for its realism, but to help isolate the effect of local traffic on each method's performance. The nonuniform destinations and long routes scenarios are more challenging and realistic, as they require the methods to cope with an abundance of non-local traffic.

We present initial proof-of-concept results in small networks and then study the same three scenarios in larger networks to show that the max-plus approach

scales well and that the qualitative differences between the methods are the same in more realistic scenarios.

For each case, we consider three different metrics: 1) *average trip waiting time* (ATWT): the total waiting time of all vehicles that have reached their destination divided by the number of such vehicles, 2) *ratio of stopped vehicles* (RSV): the fraction of all vehicles in the network that do not move in a given timestep, and 3) *total queue length* (TQL): the number of vehicles that have been generated but are still waiting to enter the network because the outbound lane of their edge node is full.

Due to lack of space, we present graphs only for ATWT. In most cases, ATWT is sufficient to determine the methods' relative performance, i.e. lower ATWT implies lower RSV and TQL. Therefore, we mention the RSV and TQL results only when they are qualitatively different from ATWT. All results are averaged over 10 independent runs.

6.1 Small Networks

Figure 2 shows the small network used for the baseline and nonuniform destinations scenarios. Each intersection allows traffic to cross from only one direction at a time. All lanes have equal length and all edge nodes have equal spawning rates (vehicles are generated with probability 0.2 per timestep). The left side of Figure 3 shows results from the baseline scenario, which have uniform destinations. As a result, much of the traffic is local and hence there is no significant performance difference between TC-1 and max-plus. TC-SBC performs worse than the other methods, which is likely due to slower learning as a result of a larger state space.

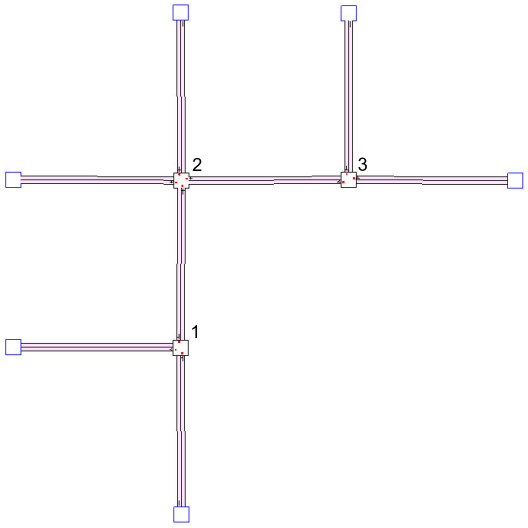

Fig. 2. The small network used in the baseline and nonuniform destinations scenarios.

The right side of Figure 3 shows results from the nonuniform destinations scenario. In this case, all traffic from intersections 1 and 3 is directed to intersection 2. Traffic from the top edge node of intersection 2 is directed to intersection 1 and traffic from the left edge node is directed to intersection 3. Consequently, there is no local traffic. This results in a dramatic performance difference between max-plus and the other two methods.

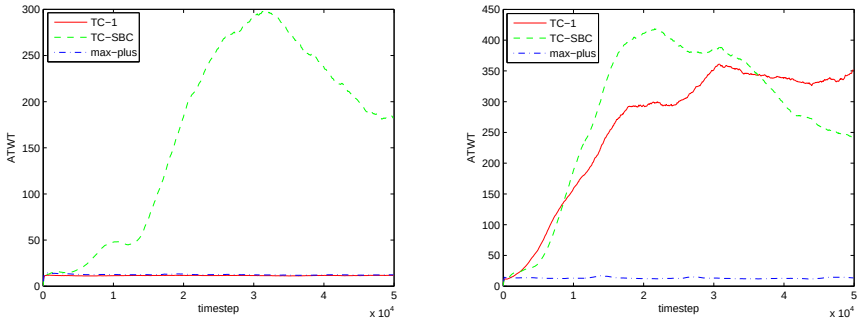

Fig. 3. Average ATWT per timestep for each method in the small network for the baseline (left) and nonuniform destinations (right) scenarios.

This result is not surprising since the lack of uniform destinations creates a clear incentive for the intersections to coordinate their actions. For example, the lane from intersection 1 to 2 is likely to become saturated, as all traffic from edge nodes connected to intersection 1 must travel through it. When such saturation occurs, it is important for the two intersections to coordinate, since allowing incoming traffic to cross intersection 1 is pointless unless intersection 2 allows that same traffic to cross in a "green wave".

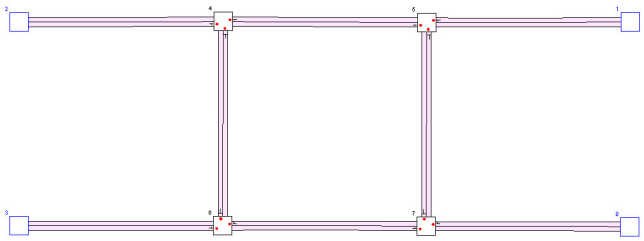

Fig. 4. The small network used in the long routes scenario.

To ensure that the performance difference between the baseline and nonuniform destinations scenarios is due to the removal of local traffic and not some other effect of nonuniform destinations, we also consider the long routes scenario. Destinations are kept uniform, but the network structure is altered such that all routes involve at least two intersections. Figure 4 shows the new network, which has a fourth intersection that makes local traffic impossible. Figure 5 shows the results from this scenario.

As before, max-plus substantially outperforms the other two methods, suggesting its advantage is due to the absence of local traffic rather than other factors. TC-1 achieves a lower ATWT than TC-SBC but actually performs much worse. In fact, TC-1's joint actions are so poor that the outbound lanes of some edge nodes become full and its TQL skyrockets. As a result, the ATWT is not updated, leading to an artificially low score. At the end of each run, TC-1 had

an average of TQL of 9259.7 while TC-SBC had only 3966.9 and max-plus had 0.0. The low quality of TC-1's joint actions becomes clear when comparing the RSV: 0.92 for TC-1, 0.55 for TC-SBC, and 0.09 for max-plus.

6.2 Large Networks

We also consider the same three scenarios in larger networks to show that the max-plus approach scales well and that the qualitative differences between the methods are the same in more realistic scenarios. Figure 6 shows the network used for the baseline and nonuniform destinations scenarios. It includes 15 agents and roads with four lanes. The left side of Figure 7 shows results from the baseline scenario, which has uniform destinations. As with the smaller network, max-plus and TC-1 perform very similarly in this scenario, though max-plus's coordination results in slightly slower learning. However, TC-SBC no longer performs worse than the other two methods, probably because the network is now large enough to incur substantial congestion. TC-SBC, thanks to its congestion bit, can cope with this occurrence better than TC-1.

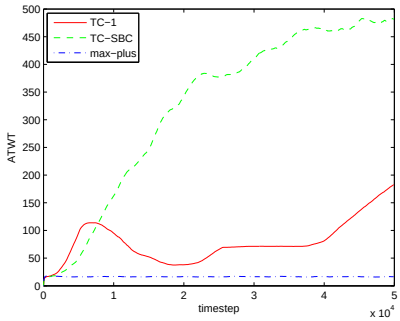

Fig. 5. Average ATWT per timestep in the small network for the long routes scenario.

The right side of Figure 7 shows results from the nonuniform destinations scenario. In this case, traffic from the top edge nodes travel only to the bottom edge nodes and vice versa. Similarly, traffic from the left edge nodes travel only to right edge nodes and vice versa. As a result, all local traffic is eliminated and max-plus performs much better than TC-1 and TC-SBC. TC-SBC performs substantially better than TC-1, as the value of its congestion bit is even greater in this scenario.

To implement the long routes scenario, we remove one edge node from the two intersections that have two edge nodes (the top and bottom right nodes in Figure 6). Traffic destinations are uniformly distributed but the new network structure ensures that no local traffic occurs. The results of the long routes scenario are shown in Figure 8. As before, max-plus substantially outperforms the other two methods, confirming that its advantage is due to the absence of local traffic rather than other factors.

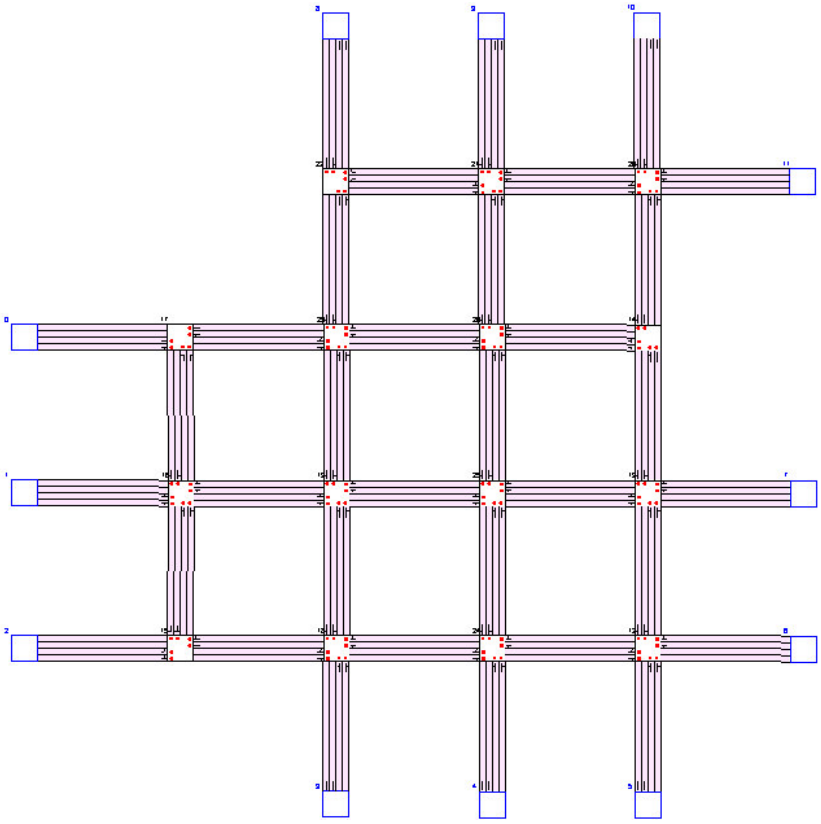

Fig. 6. The large network used in the baseline and nonuniform destinations scenarios.

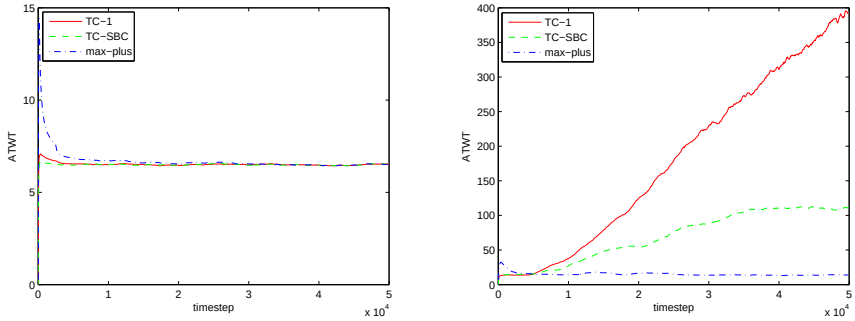

Fig. 7. Average ATWT per timestep for each method in the large network for the baseline (left) and nonuniform destinations (right) scenarios.

7 Discussion

The experiments presented above demonstrate a strong correlation between the amount of local traffic and the value of coordinated learning. The max-plus method consistently outperforms both non-coordinated methods in each scenario where local traffic has been eliminated. Hence, these results help explain under what circumstances coordinated methods can be expected to perform better. More specifically, they confirm the hypothesis that, under highly saturated conditions, coordination is beneficial when the amount of local traffic is small.

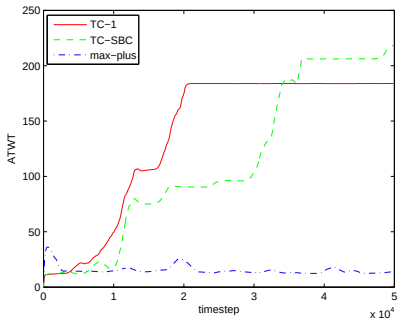

Fig. 8. Average ATWT per timestep for each method in the long routes scenario.

Even when there is substantial local traffic, the max-plus method achieves the same performance as the alternatives, though it learns more slowly. Hence, this method appears to be substantially more robust, as it can perform well in a much broader range of scenarios.

By testing both small and large networks, the results also demonstrate that max-plus is practical in realistic settings. While max-plus has succeeded in small applications before [10], this paper presents its first application to a large-scale problem. In fact, in the scenarios without local traffic, the performance gap between max-plus and the other methods was consistently larger in the big networks than the small ones. In other words, as the number of agents in the system grows, the need for coordination increases. This property makes the max-plus approach particularly attractive for solving large problems with complex networks and numerous agents.

Finally, these results also provide additional confirmation that max-plus can perform well on cyclic graphs. The algorithm has been shown to converge only for tree-structured graphs, though empirical evidence suggests it also excels on small cyclic graphs [10]. The results presented in this paper show that this performance also occurs in larger graphs, even if they are not tree-structured.

8 Future Work

There are several ways in which the work presented this paper could be extended or improved. First, the algorithm could be augmented to automatically discover the best coordination graph for the problem. We currently use fixed coordination graphs with dependencies only between intersections connected by a road. In some cases, other coordination graphs could lead to better performance. Optimization methods such as genetic algorithms could potentially be used to search for the best coordination graph for a given problem.

Second, max-plus could be implemented in a distributed fashion. The current implementation is centralized and uses iterations. In each iteration, an agent sends messages to its neighbors in a predefined order. The same functionality could be achieved with a distributed implementation where each agent sends an updated message as soon as it receives a new message from a neighbor. Since messages would be sent in parallel, computational savings would occur. However, such an implementation would require the development of protocols and other functionality not presently supported by the simulator.

Third, vehicle route selection could be adapted on-line, i.e. traffic controllers and vehicles could coordinate their behavior based on real-time traffic conditions. This functionality could avoid bottlenecks by distributing traffic more wisely over the network. Such an approach may be difficult to implement in practice since it requires vehicles to cooperate and communicate with the traffic controllers. However, previous work [14] has generated promising initial results for such an approach.

Fourth, several simplifying assumptions could be removed from the simulator. The environment is currently fully observable and stationary. Communication costs are not modeled. Many real-world factors such as weather, pedestrian behavior, vehicle accidents, illegal parking, etc. are not considered. Overall, the current simulation does exhibit most of the crucial characteristics that make urban traffic control difficult. Nonetheless, future work should focus on constructing even more realistic environments and developing the algorithmic extensions necessary to tackle them.

9 Conclusions

This paper presents a novel method for learning efficient urban traffic controllers. Previous work used multiagent reinforcement learning but the agents selected only locally optimal actions without coordinating their behavior. This paper extends this approach to include explicit coordination between neighboring traffic lights. Coordination is achieved using the max-plus algorithm, which estimates the optimal joint action by sending locally optimized messages among connected agents. This paper presents the first application of max-plus to a large-scale problem and thus verifies its efficacy in realistic settings. Empirical results on both large and small traffic networks demonstrate that max-plus performs well on cyclic graphs, though it has been proven to converge only for tree-structured graphs. Furthermore, the results provide a new understanding of the properties a traffic network must have for such coordination to be beneficial and show that max-plus outperforms previous methods on networks that possess those properties.

References

1. B. Abdulhai, et al. Reinforcement Learning for True Adaptive Traffic Signal Control. In *ASCE Journal of Transportation Engineering*, 129(3): pg. 278-285, 2003.
2. A.W. Moore and C.G. Atkenson, Prioritized Sweeping: Reinforcement Learning with less data and less time, in *Machine Learning*, 13:103-130, 1993.
3. B. Bakker, M. Steingrover, R. Schouten, E. Nijhuis and L. Kester, Cooperative multi-agent reinforcement learning of traffic lights. In *Proceedings of the Workshop on Cooperative Multi-Agent Learning, European Conference on Machine Learning*, ECML'05, 2005.
4. R. E. Bellman, Dynamic Programming Princeton University Press, Princeton, NJ, 1957.
5. S. Chiu. Adaptive Traffic Signal Control Using Fuzzy Logic. in *Proceedings of the IEEE Intelligent Vehicles Symposium*, pg. 98-107, 1992.
6. C. Crick and A. Pfeffer. Loopy belief propagation as a basis for communication in sensor networks. In *Proceedings of Uncertainty in Artificial Intelligence (UAI)*, 2003.
7. Foy M. D., R. F. Benekohal, D. E. Goldberg. Signal timing determination using genetic algorithms. In *Transportation Research* Record No. 1365, pp. 108-115.
8. F. R. Kschischang, B. J. Frey, and H.A. Loeliger. Factor graphs and the sum-product algorithm. In *IEEE Transactions on Information Theory*, 47:498519, 2001.
9. C. Guestrin, M.G. Lagoudakis, and R. Parr. Coordinated reinforcement learning. In *Proceedings Nineteenth International Conference on Machine Learning*, pg 227-234, 2002.
10. Jelle R. Kok and N. Vlassis. Collaborative Multiagent Reinforcement Learning by Payoff Propagation, in *J. Mach. Learn. Res.*, volume 7, pg. 1789-1828, 2006.
11. Ma Shoufeng, et al. Agent-based learning control method for urban traffic signal of single intersection. In *Journal of Systems Engineering*,17(6): pg. 526-530, 2002.
12. J. C. Spall, and D.C. Chin. Traffic-Responsive Signal Timing for System-wide Traffic Control. in *Transportation Research Part C: Emerging Technologies*, 5(3): pg. 153-163, 1997.
13. Thorpe, T. L. and Andersson, C.. Traffic light control using sarsa with three state representations. *Technical report*, IBM corporation,1996.
14. M. Wiering, Multi-Agent Reinforcement Learning for Traffic Light Control, in *Proc. 17th International Conf. on Machine Learning*,pg. 1151-1158, 2000.
15. N. Vlassis. A Concise Introduction to Multiagent Systems and Distributed Artificial Intelligence. Synthesis Lectures in Artificial Intelligence and Machine Learning, Morgan & Claypool Publishers, 2007.
16. J. S. Yedidia, W. T. Freeman, and Y. Weiss. Understanding belief propagation and its generalizations. In *Exploring Artificial Intelligence in the New Millennium*, chapter 8, pg. 239269, 2003.
17. N. L. Zhang and D. Poole. Exploiting causal independence in Bayesian network inference. In *Journal of Artificial Intelligence Research*, 5:301328, 1996.