Controlling Complex Dynamic Transportation Systems: Development and Adaptation of a Novel Distributed Cooperative Multi-Agent Learning Technique

A Dissertation Presented for the

Doctor of Philosophy

Degree

The University of Tennessee, Knoxville

Russell Thomas Graves

May 2024

Abstract

Intelligent transportation systems continue to increase complexity, scale, and scope as more devices contain embedded compute. Cooperation among vehicles, intersections, and other members of the greater traffic ecosystem at a system-of-systems level is critical to improving the efficiency of the multi-billion-dollar asset that is the U.S. roadway infrastructure.

This work introduces a negotiations strategy among multi-agent reinforcement learning agents and applies this to both traffic signal control and supervisory control of vehicle platooning. The traffic signal control implementation builds off of many prior research thrusts, and was shown to improve vehicle throughput by an average of $671 veh/hr$ over actuated traffic under static arrival conditions. Additionally, the negotiations strategy saved $9.8 sec$ and a maximum of $14.3 sec$ in total travel time for a probe vehicle traversing the intersection system under static conditions. The realized efficiency increases resulted in an emissions reduction of $809 kg/hr$.

In the case of platooning, the presented framework departs from the bulk of research, which tends to focus on the safety-critical and fine control of platooned vehicle motion; instead, this work chooses to approach platooning from a supervisory angle. Under the guidance of negotiated supervisory control, the maximum observed fuel savings was 20.8% in a low-traffic-density scenario over the nominal case where no vehicles are directed to platoon and closer to 7.2% when traffic is calibrated to match US highway 101. In each of these cases, the differences in mean speed between legacy driven vehicles and the vehicles supervised by the negotiations was $< 5 m/s$.

While the underlying representations for each of these systems is necessarily different. The results suggest that approaching the increasingly well-instrumented and controlled

network of roadways with supervisory level controls yields an improvement in performance. The extension of this approach and future works are also addressed.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

As of 2019, transportation accounted for roughly 29% of U.S. greenhouse gas emissions, local governments spend as much as $203 billion on roads and highways, and US business have spent as much as $1.04 trillion on transportation [7, urb, 18, 49]. These costs have drove combustion engine research in the past, battery and motor technology to where it is today, and Intelligent Transportation Systems (ITS) alongside both of these.

ITS is a broad term covering areas such as arterial and freeway, freight, transit, incident, emergency, and other traffic/transportation management systems [65]. Individual research in ITS may focus on specific activities such as: traffic signal timing, vehicle autonomy, vehicle platooning, traffic sensing, etc.. Since the 1970s ITS solutions have been developed alongside improving methods and algorithms for managing, accessing, and analyzing ever more available and increasingly large pools of transportation data [91]. Much of this data comes from sensors and compute hardware which are increasingly smaller, more powerful, and reliable enough to embed into infrastructure components and vehicles ([22, 26, 36, 84, 68]). The scope of this effort to instrument and traffic elements is even extending to rural areas using low-cost webcams([66]).

As the amount of available data, sensors, and controllers increases, so too does the number of machine learning methods which seek to exploit the wealth of data in order to improve understanding and performance of transportation networks [87]. Machine learning (ML) is an area of study which seeks to leverage data in order to improve the performance of tasks [59]. ML algorithms typically ingest data, referred to as "training data" and, through

some algorithm, "learn" to perform a task e.g. classification, regression, action selection, clustering, etc. for which it was not explicitly programmed. Once these algorithms have ingested the training data, they are referred to as "trained", and the individual "trained" machine learning decision-maker or predictor will be called the "agent". When a system is very complex or data is of high dimension, ML techniques are exceptionally good at identifying subtle relationships and patterns in the data which would otherwise be impossible to extract by hand and hand-craft a set of rules to exploit. These algorithms have enabled ITS solutions to exercise more precise control over individual elements e.g. traffic signal controllers and vehicles.

Rather than dictate control to each element of an ITS system as the number of intelligent elements increases, it can be helpful to decompose the problem into heirarchical levels of fidelity for analysis, estimation, and control. This is similar in intention to the idea of system architecture views and aligns with the systems engineering approaches pursued by the United States Department of Transportation (USDOT) and others [39, 8]. A great deal of effort has been applied to coordinating among signals or platooning vehicles with low-level control. The strategy and applications presented in this work should fill a gap by providing a novel means of coordinating between multiple heterogeneous agents at a supervisory level. This level of control may dictate, in broader strokes, the behaviors desired from individuals or groups of individuals, and it strategically leaves safety critical or low-level control to other processes. The proposed method borrows and adapts recent and ongoing developments in artificial intelligence and machine learning to craft the supervisory control algorithm, and these specific ITS applications serve as applications.

In the case of ITS, in addition to many real-world applications, multiple actors are present in systems at the same time. Often, control of the system is similarly distributed across a number of decision makers or devices. These multi-agent systems (MAS) focus on the interactions between agents in the environment, their decisions, and the interactions between them [62]. Multi-agent learning (MAL) is a class of ML techniques which is specifically applied to MAS. In ITS applications, the agents in these MAS are mostly considered cooperative. A common goal is shared between each agent; this is typically related to energy consumption, flow rate, flow stability, or space.

One of the principle challenges of MAS is dealing with or preventing state explosion [75]. Many of the typical proofs for convergence for RL algorithms rely on sufficient visitations to each "state" in an environment [74]. If the complexity of the environment increases or the number of agents in an environment increases, it becomes increasingly difficult to satisfy this criteria. In this work, an MAL solution strategy is presented which incorporates data exchange between agents and dyna-style learning leveraging a model of the environment. This method incorporates inspiration from both classic game theory approaches such as Brown's fictitious play and more modern concepts such as the Pseudo Dyna-Q presented by Zou et al. in order to provide a solution for a MAS [92, 27]. In the following chapters, the method is adapted to two ITS applications, traffic signal control and vehicle platooning.

1.1 Literature review

1.1.1 Machine Learning

Machine learning is a subset of artifical intelligence which can be further subdivided into three general categories:

1. Unsupervised learning
2. Supervised learning
3. Reinforcement learning

which are clearly defined in literature [12, 52, 59, 21]. Supervised and unsupervised learning generally deal with prediction or classification tasks. Supervised learning, specifically, relies on "labels" attached to the training data which enable the agent to determine if it has correctly predicted, classified, or acted. In the unsupervised case, these labels are not available, and the agents leverage the properties of the training data in order to group or cluster individual samples based on similarities between them. The last category, Reinforcement learning (RL), focuses on driving agent training through trial and error [72]. Unlike supervised learning, the agent often is making decisions in a model of the domain or, "environment", in which the agent will operate, instead of assessing samples from a curated

set of training data. The environment model provides feedback to the agent in the form of a reward or penalty based on the actions taken or decisions made. The reward or penalty can be carefully crafted in order to inspire the agent to learn a set of behaviors by finding an optimal method for maximizing rewards or minimizing penalties.

1.1.2 Supervised and Unsupervised Learners

Supervised learning techniques rely on labeled training data to evolve a method for extracting specific information from a set of inputs [43]. In other words, a human expert must first manually provide "correct" responses for the agent to observe. During training, a supervised learner makes a prediction based on its current internal logic. The agent's output is compared against the labeled data, and identified as a correct or incorrect response. The difference between agent prediction and actual result provides feedback which is used to update the learner. Examples of supervised learners are linear regressions, support vector machines (SVM), naive Bayes classification (NB), decision trees, and most neural networks (NN) (including deepnets). The principle task supplied to each of these algorithms is to classify objects.

Unsupervised learning takes a different approach to accessing and interpreting data. Rather than rely on labels or subject matter experts (SMEs), algorithms in this category repeatedly comb through sample sets and measure relationships between individual observations in the environment [16]. Over time, the ML algorithm settles on decision regarding the "nearness" of different data points. The end result is a learner which is capable of taking a set of feature inputs and binning an associated sample into classes or categories based on these learned relations. Examples of ML algorithms in this category are self organizing maps (SOM), specific NN configurations, expectation-maximization algorithms, and k-means clustering. Rather than focusing on specific classification, these methods establish classes, clusters, and patterns which may otherwise have been looked over [38]. The algorithms and methods focused on in the body of this work leverage primarily algorithms in the third category, reinforcement learning (RL).

Reinforcement Learning

RL takes a different approach to ML than the above. Rather than taking feedback in terms of correct or incorrect, a reinforcement learner relies on a reward function to drive changes in behavior. Actions, decisions, classifications, etc. are transformed into a penalties or utilities which the learning agent compares to determine a best course of action. A seminal reference for RL is Sutton and Barto [72]. Popular RL algorithms include state-action-reward-state-action (SARSA), Q-learning, Monte-Carlo, temporal difference, and dynamic programming. Both SARSA and Q-learning are of principle interest to this body of work. These two algorithms are predicated upon the establishment of a discrete environment consisting of finite states in which an autonomous agent operates by exercising a set of available actions [67, 78, 33]. SARSA is the on-policy equivalent of Q-learning, sharing a comparable algorithm for updating the values of states and actions, and differing only in that it remains the same policy for at least two decisions, hence the name, $Q(s, a, r, s', a')$. Value updates are done at the end of this sequence. Q-learning is "off policy" and the values for states and actions, $Q(s, a)$ are updated at each decision [55, 78].

A subset of RL of particular interest are the "Dyna" style strategies. In these algorithms, an underlying model is kept of the environment, typically informed by visitation to individual states or by some other approximation [46, 92]. This underlying model enables off-policy exploration and learning where an agent may take a number of simulated actions and, using the model, estimate the results. Researchers have explored improving the speed at which these models converge through clever manipulation or prioritization of the simulated exploration process [73]. Data exchange to accurately update the underlying model has also been investigated in an application of Dyna-Q learning for MAS [35].

1.1.3 Multi-Agent Learning

Often, teams of individual workers are necessary or preferable to single-agent solutions for solving a task. Or, in the case of ITS, multiple vehicles or infrastructure components are an unavoidable fact. Thus, MAL research progressed logically to adapt ML to these scenarios. Several surveys have been conducted of multi-agent learning (MAL) techniques [62]. Panait

et al. assert two types of multi-agent learning: team learning and concurrent learning. In the former, a single agent learns how to control every member of a team simultaneously. In the latter category, multiple learners control individual team members.

In this sense, "team learning" could be viewed as "centralized control" and "concurrent learning" would be "decentralized". Decentralization is a strategy by which a complex control structure is broken up or decomposed into smaller components with the goal of distributing computational load [13, 70]. This is appealing for systems such as transportation networks which require simultaneous control of a large number of signals, vehicles, and other roadway objects. A challenge for MAL is the desire to drive the system towards a team optimal rather than individual components operating for themselves. This brings up the concept of a Nash or other game theoretic equilibrium which agents in a MAS may reach rather than achieving some shared maximum [20, 61]. While individual agents may not necessarily be in strict competition, lack of global perspective present in a centralized solution makes a global optimal solution more difficult in the decentralized case.Examples of efforts to overcome this difficulty include the Max-Plus algorithm. This was developed as a solution to MAL demonstrated that under certain restrictions, the propagation of utility values through an environment allows a set of agents' actions to converge ([42]). This converged action set, although sometimes sub-optimal for an individual agent, improves the performance of the total system.

The most recent lines of work have investigated integrating deep learning models into the Multi-Agent Reinforcement Learning (MARL) space [28, 77, 14, 32]. In these models, the deep neural networks automatically extract features necessary for making decisions and estimating long term rewards via back propagation of errors and intelligent design of the learning process. These methods rely heavily on data to make up for the lack of hand-crafted features or elements. Given sufficient information regarding the system and perhaps the global reward, a deep multi-agent reinforcement learning algorithm could extract automatically the features necessary to produce a more globally optimal set of behaviors.

The algorithm presented in this work does, however, not rely on deep neural networks. Although individual MAS are likely unique, a group evaluating both deep neural network-based and traditional RL for possible improvements in traffic signal control concluded that

simpler methods were likely as effective and less costly to implement than more complicated machine learning strategies ([29, 30]). Instead, simpler or more well-established models and estimates provide the underlying intelligence for the presented Dyna-style RL algorithm. These simpler methods typically have the added benefit of increased explainability which is crucial for safety-critical applications.

1.1.4 Traffic Signal Control

One of the first widely accepting timing protocols was developed by Webster in 1958 ([81]). Computation and controls for complex systems were developed in parallel with traffic and transportation control structures through the 1980's resulting in dynamic traffic management experiment which extended the concept of signal timing to incorporate coordination between upstream and downstream intersections. Systems like SCOOT and SCATS (GLIDE) represent real-world applications of such techniques in cities like Sydney and throughout the United Kingdom ([34, 71]). These applications of ITS based signal timing procedures introduced a new era in traffic control and infrastructure management driven by the increasing availability of computational power. This thrust to leverage new computing technology to control traffic signal networks has been well cataloged by several groups which give overviews of RL, MAL, and general AI-based solutions for signal timing ([17, 89, 85, 62]).

Some works directly model signals as M/M/1 queues and employ solution techniques for optimizing requests for service in sequences of queues ([86]). Other approaches draw inspiration from fluid dynamics and equate the problem of traffic management to back-pressure routing ([63, 82]). In the vein of machine learning, at least one study explored junction tree learning, and another has applied pure deep learning to the problem of signal control ([90, 76]). Other recent works have taken a blended approach, leveraging emerging deep-network supplemented RL strategies ([11, 47]).

1.1.5 Platooning

Platooning is a method to improve the fuel efficiency of multiple vehicles by driving in close distances to one another and taking advantage of reduced wind resistance from trailing

another vehicle [40]. This includes the benefit of traffic efficiency, where several vehicles driving within close proximity to one another results in more space on the road for additional vehicles or smoother flow [88]. In order to achieve a vehicle platoon, it is necessary for autonomous systems to sense or communicate with one another in order to avoid collisions while closing the gap between platoon members. There have been demonstrations of both pure longitudinal control and others including lateral control including SARTRE, PATH, Energy ITS [19].

Other methods for grouping or creating platoons of vehicle include Adaptive Cruise Control (ACC), where a vehicle will match the speed of the vehicle directly in front of itself [83]. Vehicles with these systems have been on the market globally since 1995, and an ISO standard was created in 2002 [Norm]. Enabling vehicles to communicate beyond their immediate neighbors, referred to as Cooperative Adaptive Cruise Control (CACC), allows for smoother vehicle velocity profiles while maintaining sufficiently small gaps between following cars [57, 64]. Although the leading vehicle was controlled completely by a legacy driver and was not autonomous, all trailing vehicles had CACC and were manned by professional truck drivers as a safety precaution. An added safety parameter that CACC can utilize, as was the case in the report, is if a vehicle is unable to establish communication with any other vehicle, then it can default into ACC until such a time as connection can be re-established [50].

1.1.6 Systems of Systems

If the dream of a fully autonomous transportation future is realized, then ITS would likely qualify as a "System of Systems" (SoS) [51, 23]. Addressing ITS as a SoS and crafting architecture definitions to facilitate the development of these systems is certainly not a new concept [53]. However, even Maier admitted that no architecture for ITS satisfied both the operational independence and managerial independence requirements at the time due to the myriad of conflicting, simultaneous goals held by the stakeholders in the broader transportation community: private individuals, governments, and corporate interests [51]. With the added wrinkle of autonomous vehicles, the heterogeneous human and AI ITS landscape is likely more aptly categorized as a "Complex System" [69, 80]. In these cases,

the focus is often on emergent behaviors as a result of changes in the constituent systems or control logic, and the goal is to make incremental improvements in the operation of the system.

This work specifically addressed the complexity problem through key points of decomposition, targeted simplification, and satisficing when appropriate. Other efforts to architect appropriate inter-vehicle and inter-infrastructure (V2X) have similarly looked to decompose the full system of systems into smaller, more manageable segments [58]. Platooning and signal control were approached from a supervisory level, leaving specific stability concerns and safety critical aspects aside. Focusing on higher-level guidance for platooning vehicles afforded a similar multi-second time scale which aligned well with decisions regarding the next traffic signal cycle. The presented algorithms treated the platooning vehicles as probes to glean information regarding local traffic patterns, thereby reducing the chaos of human decision making to a more digestible set of summary statistics. And, critically, decisions of the agents themselves were summarized and exchanged in a simplified likelihood set which, while most likely leading to less-than-optimal results, enables fast and lightweight interaction between collaborating elements. All of these decisions make possible the adaptation of the presented methods to collaborative supervisory control of this complex system.

Chapter 2

Negotiations

The issue underlying many ITS and similar applications is complexity. In the past, many of the systems operating in the same environment were manually controlled or simply existed as background elements with respect to control problems. When designing individual automated pieces or elements, it was (and still may be) possible to summarize the impact of these uncontrolled elements through a series of simplifying assumptions. However, systems and systems-of-systems have become more automated, intelligent, and connected. As soon as a manual or background element acquires a micro-controller, processor, etc., it becomes a source of data or an element which could be used to effect some change in the larger system. Modern designs sought to leverage these new vectors rather than continue to address the control of individual elements in a vacuum. ML has been a key tool in these processes; it both provided a means to more efficiently analyze the data from these systems, and it enables the design of complex supervisory and other controllers which ingest, process, and use high dimensional system data to more efficiently direct these complex systems. The algorithm introduced in this work focuses on the supervisory control of similar complex environments where multiple autonomous decision-making agents exist alongside uncontrolled background actors and elements. The proposed method attempts to address these systems by:

1. Distribution
2. Decomposition
3. Communication

where each agent participates in a distributed MARL problem, maintains its own internal decomposed model of the environment, and communicates a simplified summary of intent to other actors in the ecosystem.

MARL problems often inherently suffer more from curse of dimensionality and the state exploration tradeoff by introducing inherent non-stationarity, difficulties in designing a reward function, and confusion when assigning credit for actions[24]. Executing a centralized strategy to direct all agents in a MAS can mitigate the issue of assigning credit, but it often comes at the cost of an increase in the number of states. Conversely, a distributed strategy can help decrease the number of states, but this makes the issue of credit assignment and, by extension, determining the long term value of an individual agent's actions more difficult. Of the two options, this work focuses on the dimensionality and exploration problems.

In order to address these two issues, an underlying model of the environment was crafted according to prior information. This model enables agents to evaluate the impact of their actions through sets of offline simulated decision cycles. At regular intervals, each agent broadcasts a set of probabilities, referred to as "propensities", which provide information regarding its current mixed strategy. In the applications of the algorithm which follow, the structure of the underlying model changes. The goal of this algorithm is not to dictate the nature of this model, and it should be noted that the core premise of the negotiations strategy could be adapted to use other methods for the decision-making process or planning processes. For example, a data-driven method e.g. deep learning, could be substituted for the underlying model, or something like deep reinforcement learning could leverage the propensities as part of the agent's observation.

In this chapter, the core concepts of Q-learning and the extension this work relies on, Dyna-Q, are introduced. Next, the concept of fictitious play and its relation to this work is discussed. Finally, the negotiations algorithm is defined and the convergence considerations are addressed along with some simple examples in well-known RL environments.

2.1 Markov Games

Markov games or "stochastic games" are a generalization of Markov Decision Processes (MDPs) to multiple interacting decision makers. MDPs are formally defined by the 5-tuple of state, action , reward, transition probability, and discount factor, $\{S, A, R, T, \gamma\}$. The agents operating in this framework take actions to facilitate transitions between states with the intent of maximizing some perceived reward. The addition of more than one agent in a shared environment makes this a "Markov Game". The agents may act simultaneously or in a sequence, and at each time step, a payoff is observed by each agents. This structure is commonly used for multi-agent reinforcement learning.

In order for a Markovian model to be valid, an agent's current state must be sufficient for determining the likelihood of transitioning to another state in the next time instant. Satisfying this assumption presents a significant challenge for any real-world applications of Markovian solution techniques. The environment is often non-stationary which results in an agent's past states significantly altering the current transition probabilities. Additionally, MAS typically violate this assumption when cast as MDPs. In these cases, the previous states and actions of each agent in the environment impact the transition probabilities of surrounding agents. To satisfy the stationarity environmental requirement, the problem must be carefully framed and may only be valid for a given time period or specific window of operation.

2.2 Dyna-Q Learning

Dyna-Q learning is a model-based variation of Q-learning. Q-learning is an RL solution method characterized by a feedback loop which evaluates actions coupled to a state ([79]). These state-action values, $Q(s, a)$, are observed by the RL agent and are updated according to Eqn.2.1 [72].

$$
\begin{aligned}
Q(s,a) =& (1-\alpha)Q(s,a) + \alpha(R + \gamma(max_a(Q(s',a)))) \\
Q =& \text{ Station-action value or utility} \\
s =& \text{Current state} \\
s' =& \text{Resultant state} \\
a =& \text{Action} \\
\alpha =& \text{Learning rate} \\
\gamma =& \text{Discount factor}
\end{aligned}
\tag{2.1}
$$

In the case of Dyna-Q, the agent stores an additional set of state-action values, M, which act as a model for the environment. The agent can simulate experience steps, drawing from past observations to provide further updates. In a traditional Dyna-Q implementation, Alg.1, as with many RL solution techniques, the agent's understanding of the environment relies on sufficient visits to each state. This assumes that the underlying model M begins from a random or unknown state and is only filled in each real observation.

Initialize $Q(s,a)$ and $M(s,a)$ for all $s \in S$ and $a \in A(s)$
for (*all time*)
  S $\longleftarrow$ current state
  A$\longleftarrow$ ϵ-greedy(S,Q)
  Execute action, A, observe reward, R, and state, S'
  Q(S,A)$\longleftarrow$Q(S,A)+$\alpha[R + \gamma \max_a Q(S',a) - Q(S,A)]$
  $M(S,A)$ $\longleftarrow$R,S'
  repeat
    S $\longleftarrow$ random previously observed state
    A$\longleftarrow$ random action previously taken in S
    R,S'$\longleftarrow$M(S,A)
    Q(S,A)$\longleftarrow$Q(S,A)+$\alpha[R + \gamma \max_a Q(S',a) - Q(S,A)]$
  until *n times*;

Algorithm 1: Dyna-Q Learning

In this configuration, synthetic experience can only be generated when selecting from the pool of previously visited states and actions. If the environment is unknown, then each state must be visited at least once in order to evaluate its efficacy. One of the key tenants of many RL solutions is sufficient visitation to each state. This relies on the underlying exploration

mechanic to drive the agent to new states in spite of perceived detriments. In many cases this is simply accomplished through an ϵ-greedy action selection policy. In these strategies, the agents select the maximally beneficial action with probability ϵ, and at all other times they randomly choose from the sub-optimal or unknown actions. This exploration is the process by which the artificial intelligence takes novel actions and visits previously unexplored states-action pairs. However, as the simulation proceeds, the RL agent should explore less of their environment and focus on exploiting the information it has acquired to make optimal decisions. Epsilon-greedy action selection provides a mechanism for encouraging this early exploration and later exploitation. In this variant, the initial probability ratio is slowly decreased such that an agent will either take the calculated optimal action more as time progresses. This aides in preventing the algorithm from falling into a local optima, and allows for improved performance in environments where the state transition probabilities are assumed to be static.

2.3 Fictitious Play

In Browne's Fictitious Play the convergence to a Nash solution for two agents was pursued via repeated simulated games in a known environment where one agent acts while the other agent is assumed to play a mixed strategy given by the propensity, Y [20]. In fictitious play, each actor, player 1 and player 2, has a state-action space represented by an (n x m) matrix. Player 1 has pure strategies $i \in N = 1, 2, ..., n$ and player 2 has pure strategies $j \in M = 12, ..., m$. A and B are the payoff matrices for players 1 and 2 respectively. Each player's expected payoff is calculated leveraging the mixed strategy of the other player e.g. for player 2's mixed strategy y, player 1's payoff is $(Ay)_i$. In the original implementation, the beliefs of each player regarding the others' strategy is given by Eqn.2.2:

$$\begin{aligned} x(t) &= 1/t \sum is \\ y(t) &= 1/t \sum js \end{aligned} \tag{2.2}$$

Where is and js are the set of observed plays from each player. In that way, the mixed strategy belief is simply the observed prior for the other player. These players, x and y,

update their beliefs recursively and play a pure strategy which is optimal for their current beliefs. The pure strategy may change from t to time $t+1$,

2.4 Negotiations Strategy

The exploration strategy employed by Dyna-Q assumes that the environment in which the agent operates is unknown and likely stationary. Indeed much of the RL literature focuses on leveraging either data-driven methods, such as deep learning, or exhaustive exploration of the environment in order to produce a solution. However, in many applications for MARL, reasonable models of the environment exist. In the case ITS, driver and traffic models have been studied since the 1950's and provide an understanding for the environment given some basic information about the traffic conditions [41, 56, 31]. The Negotiations strategy begins with the establishment of a model of the environment. For example, in the case of platooning, the model may be crafted based on the fundamental diagrams of traffic flow along a link. Rather than the Dyna-Q update requiring visited states, the agent may use this model to simulate experience in non-visited states. This takes inspiration from the strategy employed by Zou et al.[92]. So long as the model is representative of the system at hand, the pseudo-experience produced by sampling from non-visited states serves to both train the agents, and the feedback from the results of these actions can be used to further refine or adapt the model to better reflect conditions local to an individual agent.

In addition to enabling agents to generate synthetic experience from unvisited states, each agent communicates its strategy to neighboring agents. This process is similar to Brown's fictitious play. In this case, the Pseudo Dyna-Q (PDQ) model is used to create the local pure strategies, and the exchange of propensities is accomplished via short range wireless communications. The communications network was not studied in this effort, but examples of dedicated short range communications (DSRC) embedded on connected and autonomous vehicles has been widely pursued. In addition, many modern vehicles are connected to cellular or other data communications pathways which could be leveraged to perform this handshake [9]. Vehicle to infrastructure, V2I communications, can be similarly established and inter-infrastructure data exchange, I2I, can be accomplished via hard line or wireless

communications. The formulation of the negotiations strategies in this work relied on clever arrangement of the agents in the environment such that these communications and the fictitious play process can take place pairwise between sets of neighboring agents.

Another key difference between the proposed Negotiations strategy and Brown's original fictitious play lies in the calculation of the mixed strategy propensities. In the original, these were simply a mean over all of the observed actions. However, it may not be appropriate to assume that other agents' actions are always observable. The other agents in an environment may be outside of sensing range or move into and out of communication range. Additionally, the action intended may be unsuccessful in the case of a stochastic environment. Thus, the action observed might differ from the strategy. Therefore, simply calculating a prior based on actions taken and time may not be a consistently good measure of the likelihood for an agent to adopt a specific strategy. Rather than construct other agents' propensity values over time, the presented method requires each agent to track its own propensities. The propensities are calculated based on the individual agent's perceived values $Q(S, A)$ for taking actions in given states. Eqn.2.3 and Eqn.2.4 demonstrates how probabilities are calculated.

$$
\begin{aligned}
P_a^i &= \frac{\exp \sum Q(s, a_k)}{\sum \exp \sum Q(s, a_k)} \\
Q &= \text{State-Action Value} \\
a &= \text{Action} \\
s &= \text{State} \\
i &= \text{Agent Number}
\end{aligned}
\tag{2.3}
$$

$$
\begin{aligned}
P_s^i &= \sum_{k=1}^{n} \frac{Pr[s_t, a_k, s']}{P_{a_k}^i} \\
s_t &= \text{Current State} \\
s' &= \text{Resultant State} \\
n &= \text{Number of States}
\end{aligned}
\tag{2.4}
$$

These values are broadcast, and the steps of fictitious play are carried out in an environment model using action propensity information received by agents in the environment. This Negotiations algorithm, Alg.2, seeks to accomplish the agreement on a suite of actions across multiple agents in an environment where it is possible to provide a reasonable model up-front and make minor adjustments as the agent receives feedback.

Initialize Action Probabilities for Each i^{th} Agent
$\mathbb{P}_{\mathbb{A}} = \{P_A^1, P_A^2, ..., P_A^i\} = \{\frac{1}{2}, \frac{1}{2}, ..., \frac{1}{2}\}$;
Initialize State Probabilities for Each i^{th} Agent $\mathbb{P}_{\mathbb{S}} = \{P_S^1, P_S^2, ...P_S^i\} = \{\frac{1}{n}, \frac{1}{n}, ..., \frac{1}{n}\}$;
for (*Initial Decision Cycle*)
    Record initial state s_t^i;
    Select random action a_t^i;
    Enact action a_t^i;
while *Simulation is Active* **do**
    Record Previous State s_t^i;
    Record Previous Action a_t^i;
    Calculate delayed reward $r_{t+1}^i(s_{t+1}^i)$;
    Update Q-Values $Q_{t+1}(s_t^i, a_t^i)$ using equation [2.1];
    Initialize Negotiation Training;
    Initiate error value $err = Inf$;
    while *err>limit & iteration<Maxlimit* **do**
        Update Agent A_i of local probability values $\mathbb{P}_{\mathbb{A}}$,$\mathbb{P}_{\mathbb{S}}$;
        Calculate Transition Probabilities from $\mathbb{P}_{\mathbb{S}}$ $\mathbb{P}_{\mathbb{A}}$ values as in [3.2] $T(s_t^i, s_{t+1}^i, a_t^i)$;
        repeat
            Select random start state s_t^i;
            Select random action a_t^i;
            Randomly select resultant state according to Transition Model $T(s, a, s')$;
            Calculate reward;
            Update Q-Values using equation [2.1];
        until *N iterations are completed*;
        Calculate new P_S using equation [2.3];
        Calculate new P_A using equation [2.4];
        Update Q-Values $Q_{t+1}(s_t^i, a_t^i)$ using equation [2.1];
    end
    select optimal action a_t^i;
end

Algorithm 2: Negotiations Algorithm

In this process, the probabilities are initialized for each agents' states and actions at the beginning of the simulation. A set of error terms are established to track convergence of the probabilities. While the error terms are large, the underlying M rewards and a set

of transition probabilities inform the known environment for fictitious play. SARSA, Q-learning, or some alternate solution method is employed to create an optimal solution based on the environment probabilities and the mixed strategy of the neighboring player. This process terminates when either a time out or solution is reached. The algorithm simulates end states and rewards by randomly selecting these according to the transition probabilities for a randomly selected start state and action. The results of this learning process are used to update the existing Q values according Eqn.2.1. These updates Q values are converted to probabilities via Eqn.2.3 and Eqn.2.4 and rebroadcast to neighboring agents. This process is repeated until the error term for negotiations is small or, if a timeout is specified, the system may terminate before convergence between neighbors.

2.5 Convergence Considerations

The principle focus of this work does not include a study of the convergence nor any optimality guarantees for the Negotiations process. There are two iterative processes which need to converge to a solution in the Negotiations strategy. After each real-world action is taken, the fictitious play process must result in a set of constant mixed-strategy propensities which will inform the next decision taken by each agent. Additionally, the over-all Dyna-Q process should converge, providing the agents in the environment with a long-term strategy which maximizes the rewards for each agent.

Convergence of Brown's original fictitious play has been well-studied [44, 20]. Since the underlying model can only be updated via some on-policy reward observation or the inclusion of a neighboring agent's new propensity values, the convergence considerations should be identical to Brown's original fictitious play. Convergence of the mixed strategies for each agent based on these simulated games and exchanges should be guaranteed so long as the environment model matrix is non-degenerate i.e. there exists no two equal strategies from any given state. This is in the case of an alternating play (AP) process, where each agents take turns making PDQ decisions and updating their propensities. However, Berger et al. point out that, by preventing simultaneous update after some given time, simultaneous play (SP) fictitious processes also converge. The negotiations strategy takes no action to

prevent non-degenerate states. However, in terms of decision-making, this simply indicates two equal-value options. In these cases, the selection action was chosen at random from the two equivalent options remaining.

Additionally, the convergence criteria and behavior of learning processes such as Q-learning and Dyna-Q are well-understood [55]. In general, these proofs rely on the use of the guarantees for temporal difference updates to the state-action values, a decaying learning rate, and sufficient visitation to each of the states. In the case of the Negotiations strategy, the underlying PDQ model enables unlimited visitation to each of the states during off-policy iterations. If the model is inaccurate, this could cause sub-optimal performance, running afoul of one point on the deadly triad. Future work in applications for Negotiations-style algorithms should include the development and tuning, automatic and manual, of these underlying models.

2.6 Example Games

A series simple games were constructed to explore the properties of the presented strategy.

1. Chicken
2. Climbing Game
3. Grid World

2.6.1 Game of Chicken

For the game of chicken, two agents participate in a competition where the actions consist of "chicken", C, and "dare", D. In the classic game of chicken the rewards are established as in table 2.1, such that each agent receives a reward of 6 when both agents chicken and 7 if the other agent chickens while they dare.

Table 2.1: Game of Chicken Rewards

	C	D
C	(6,6)	(7,2)
D	(2,7)	(0,0)

Each agent was provided a model of the environment including the rewards defined above. To put this into a representative Markov game, 5 states including a start-state and every combination of chicken and dare outcomes, listed as $S = [CC, CD, DC, DD, S]$ was created. The game was repeated in both AP and SP formats with respect to the exchange of propensity values.

First, the underlying PDQ process was only informed using the local reward model, $[6, 2, 7, 0, 0]$ with respect to the states in S, in addition to the other player propensity values. In both the in the AP and SP processes the propensity values converge in less than 10 exchanges. The resulting likelihoods for alternating play are shown in table 2.2 and simultaneous play in table 2.3.

The stochastic nature of the PDQ experience can, very seldom, result in scenarios where a sub-optimal strategy is selected in this simple case. Analyzing a specific instance where DD is chosen, a pattern of unlikely albeit not impossible simulated experience steps is selected. For example, when simulating the other agent's action, "chicken" was predicted multiple times despite the probability to "dare" being 80.3%. Thus, instead of arriving at the Nash equilibrium, there is still some likelihood that an improbable set of PDQ steps fouls the learning process. On the other hand, the solution converges part of the time to the optimal solution without any need to broadcast more than the propensity matrix.

2.6.2 Climbing Game

The climbing game originally introduced by Claus and Boutilier is positioned as a "difficult coordination game", and is therefore interesting to explore with the Negotiations strategy [25]. The reward matrix is shown in table 2.4.

The game formulation includes a true optimal (A, A) which is surrounded on either side, (A, B) and (B, A), by large negative rewards . More than likely, the agents should start in (C, C), farthest away from the catastrophic negative rewards. A likely point of convergence is (B, B), which only requires individual exploration from the agents involved to achieve. However, settling on the true optimal is highly unlikely unless both agents choose to explore simultaneously to (A, A).

Table 2.2: Alternating Play Chicken Likelihoods

	C	D
C	24.5%	40.1%
D	35.2%	0.2%

Table 2.3: Simultaneous Play Chicken Likelihoods

	C	D
C	26.6%	36.7%
D	36.6%	0.1%

Table 2.4: Climbing Game Rewards

	A	B	C
A	(11,11)	(-30,-30)	(0,0)
B	(-30,-30)	(7,7)	(0,0)
C	(0,0)	(6,6)	(5,5)

2.6.3 Grid World

Finally, the negotiations algorithm was tested against traditional Q-learning in a modified version of grid world which closely aligns with certain adaptations of the negotiations algorithm to cooperative tasks. In this implementation of grid world, there are two agents, $A1$ and $A2$. These agents work in tandem to control the actor in the world. $A1$ controls lateral movement and $A2$ controls vertical movement. The resulting state-action space could be assembled into table, but it is easier to represent with figure 2.1.

In each time step, the robot (green circle) makes a move comprised of a combination of the actions from $A1$ and $A2$. This gives the robot "king" movement potential. The rewards are designed to encourage the agents to quickly reach the exit, as in table 2.5. Both of the upper right-hand states, denoted by the star and the red symbol, are considered absorbing and will end the episode.

2.7 Results and Discussion

One, if not the most, challenging aspect of a distributed supervisory control algorithm is achieving a truly cooperative solution. By their nature, the underlying methods explored here, Q and Dyna-Q learning, are designed to solve the reinforcement learning problem for a single agent taking greedy or ϵ-greedy actions in an environment. So, without tinkering or modifying the learning process, each agent would play a "selfish" game. Even negotiating to provide more information only served to improve the self-centered play of individual agents.

In the game of chicken, the negotiating agents converge to the optimal solution of (C, C) with a likelihood value of taking the "Chicken" action of 52.7%. This is quite close to a coin-flip with respect to the broadcast probability to the other agent. Indeed, if the rewards are adjusted even slightly from those in presented in the game of chicken referenced by this work, the agents may choose to "Dare" on the odd occasion. Since they are negotiating, the (C, D) outcome is the result, with the first agent to "Dare" being rewarded as it forces the other to choose "Chicken" to avoid catastrophe during the simulated experience steps.

Working slightly to provide some context as to the full value of the system can improve the outcome. If each agent assumes the rewards of the other are identical, and naively adds

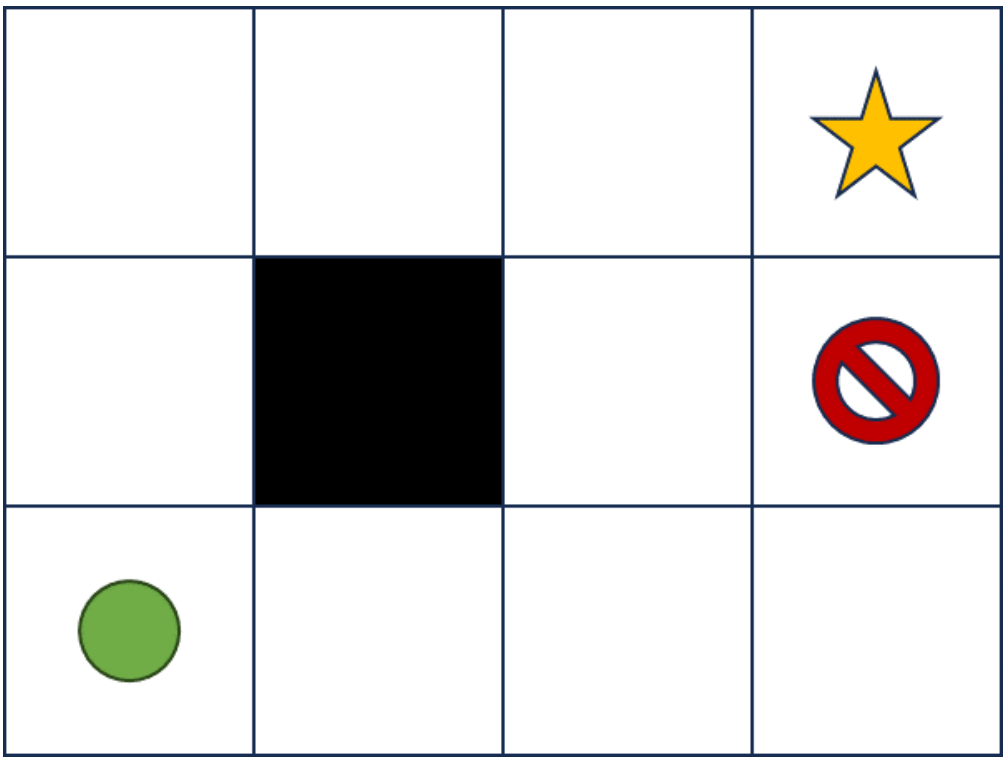


Figure 2.1: Grid world

Table 2.5: Grid World Rewards

GridWorld Rewards	*Col 1*	*Col 2*	*Col 3*	*Col 4*
Row 1	-1	-1	-1	10
Row 2	-1	N/A	-1	-5
Row 3	-1	-1	-1	-1

that value to their own reward i.e. doubling its internal reward function, the decisions are more polarized at 65.2%. The most polarizing result achieved was by fully including the expectation for each agent in the rewards. So, multiplying the perceived reward of the other agent by the likelihood of that agent taking the desired action. In this case, each agent in the game of chicken chose "Chicken" with $> 99\%$ probability. It is worth noting, however, that driving the likelihood towards 100% does not necessarily mean that the policy for a more complex game will be any more optimal.

The climbing game was noted as a "difficult" game for agents to cooperatively solve. In the configuration provided, the negotiations strategy results in a policy of (B, B). This is the result typical of other methods which are non-cooperative e.g. Q-learning and Dyna-Q learning. The broadcast likelihood values do not aide the agents in simultaneously exploring the (A, A) configuration. Even in the case where the initial model rewards match the actual rewards, the agents still find (B, B). In these instances, the likelihood of each agent to take an action other than A discourage the combined adoption of A.

However, in the case of grid world, there were some benefits to the negotiations method. Convergence of the algorithms was determined by evaluating the difference in Q-value table $Q(S, A)$ over time. For traditional Q-learning, the converged policy was shown to result in loops or incomplete pathways, e.g. figure 2.2.

The negotiated solutions generated during testing, about 100, varied between the north and south paths, but always provided a complete path to the optimal state if the greedy policy was followed. There are no claims of a guarantee that this will always be the case, but the negotiations showed some benefit in the experiments performed with the decomposed horizontal and vertical grid world.

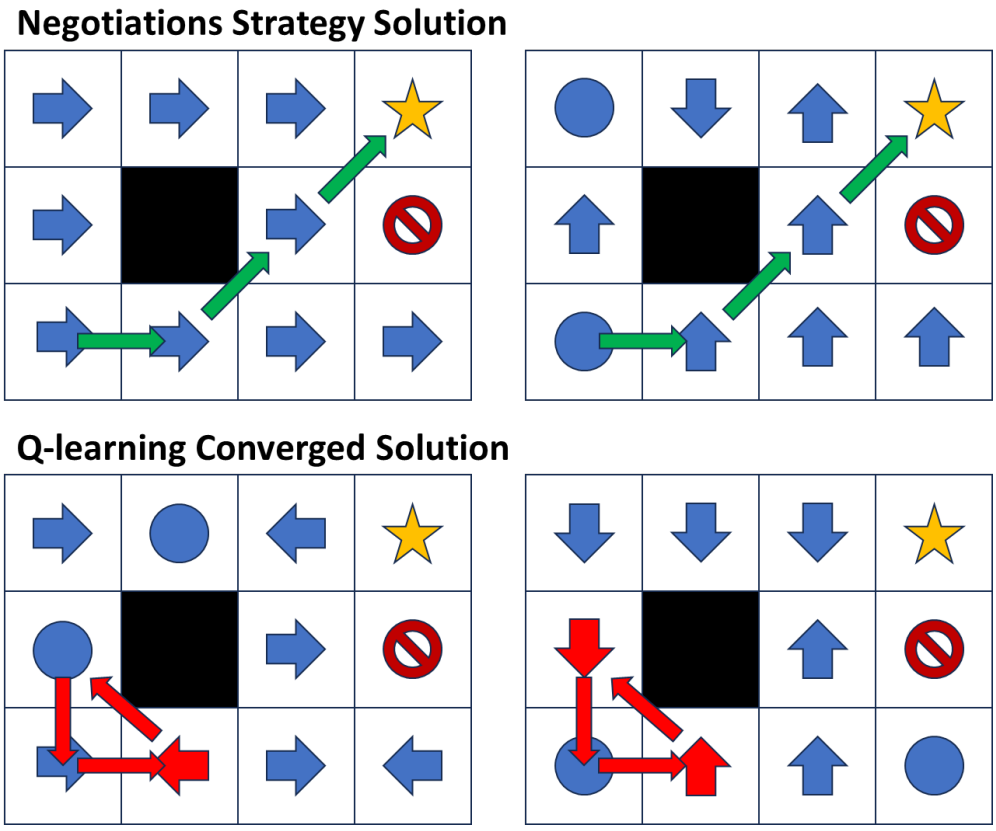


Figure 2.2: Grid world converged policies (L) Horizontal agent (R) Vertical agent

Chapter 3

Algorithm for Signal Control

The first of the two focus applications for the Negotiations technique is traffic signal control. In order to adapt this strategy to signal control, the agent and environment must be constructed appropriately. For signal control the agent is defined as an artificial intelligence which has control over either a single traffic signal and its movements. A movement, in this context, is a single direction of travel at the light, i.e. westward right turn, eastward straight-through, etc.. Each of these agents observe and exercise control over a finite length of the road surface leading up to the agent-controlled signal's location. These road segments, or "control zones", are defined as the length of road directly observed and controlled by a single agent. A simplified intersection consisting of only four straight-through movements provides an example of an agent and illustrates the control zone's movements which the agent would observe in order to determine its state (figure 3.1).

Each control zone is assumed to be of a length which allows for all vehicles within the zone to be served in a single decision-cycle. Control zone size and cycle length, C, are linked to one another. In a real-world application, an engineer may choose one of them based on any existing limitation such as sensor range or observable roadway. This is likely the most common case as modern intersections may still rely on induction loop detectors. In this case, the cycle time would need to equal the amount of time taken to serve all vehicles in the observable zone. In this work, the decision cycle is defined as 9-seconds. After the signal switches, there is a yellow phase and a brief all-red. This assumption is intended to ensure that an agent choosing to service a conflicting phase observes all appropriate

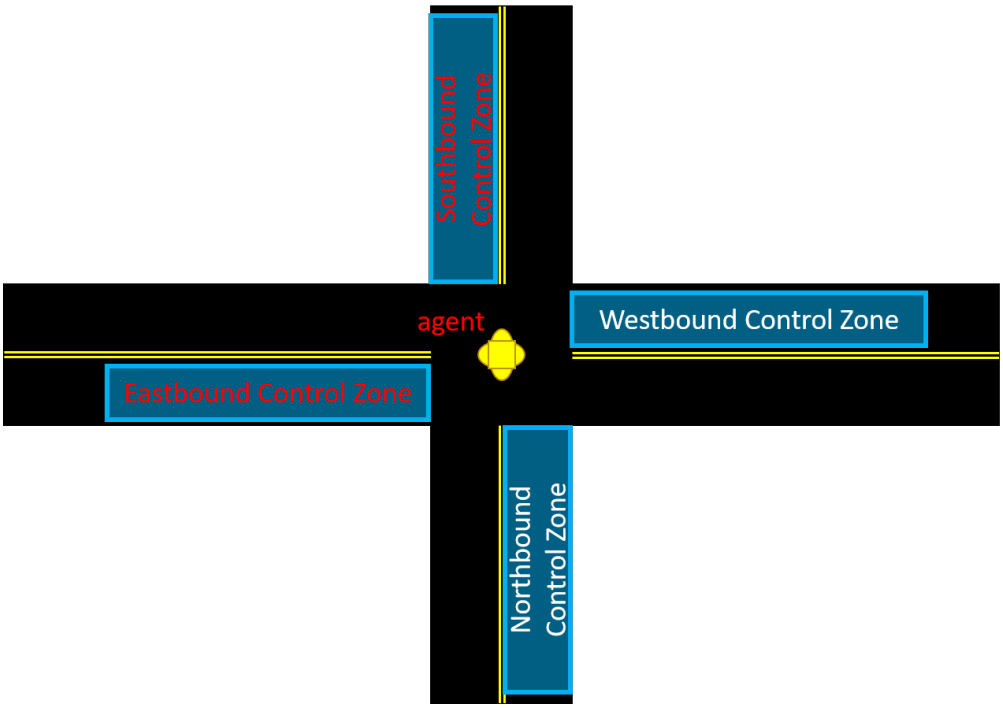


Figure 3.1: Control Zones at a Simplified Signal

safety precautions to prevent simultaneous service of conflicting movements and subsequent collisions. This assumption could be modified to fit any restrictions or regulations in a given region.

Typical modern intersections have at least eight phases which may be activated in a sequence called a cycle. In any given phase, sets of movements are serviced. Movements in each phase are designed to either completely protect the traffic in the movements, or permit certain conflicting traffic service with the expectation that vehicles will yield to one another in a specified fashion. This preset number of phases is established to reduce the overall computational complexity of the intersection. The presented method would leverage this in practice to reduce the amount of computational load and the overall scope of the state-action space. If the state-action space is too large, number of decisions required to explore it and, in doing-so, learn an optimal strategy, increases exponentially.

There is also a concern in the physical implementation of a system which leverages these control zones. If two intersections are far apart, the control zones may not be long enough to entirely bridge the gap between two signals. Lengthening the control zones, however, would violate the assumption that all vehicles will move out of a given control zone in a single decision cycle. Additionally, if a control zone is made too large, then slow traffic speeds, jam conditions, and other common traffic phenomenon may disrupt the flow of vehicles such that moving all cars to the next zone is impossible in one decision cycle. To cope with this, long sections of roadway are divided into a series of control zones, as shown in figure 3.2. These are referred to as intermediary zones. The length of these zones are assumed to be influenced by mean vehicle speed in order to allow cars in one zone passage to the next in a single decision cycle. This design assumption removes non-stationary environmental factors caused by irregular vehicle movements, speeds, and distances between traffic signals.

It is important to note that these intermediary zones are not physical in any way, and can be recreated or resized if vehicles speeds along the roadway change significantly. This means that the learned strategy will likely only be relevant for periods of time in which the traffic behavior is similar. In practice, supervisory control or some overarching method could be used to store previously learned strategies or underlying environment models and swap between them based on observed system conditions. This was not investigated, but this

Figure 3.2: Roadway broken up into artificial control zones

sort of strategy selection is commonly used with manually timed signals today for similar reasons. These ad hoc zones give each signal a method for estimating the number of cycles before vehicles leaving an upstream intersection arrive in their control zones. A real-world implementation of this would only require that the intersection agents be able to count the number of serviced vehicles. This count enables downstream traffic signal agents to estimate state probabilities and broadcast those along with their action propensity values. It also provides flexibility if sensors or probe vehicles are able to provide information regarding specific cells along the roadway between signals. These measurements could be used to increase the accuracy of an intersection's estimate for incoming vehicles.

3.1 State Definition

The state set for this work is simplified by the assumption that a control zone may be emptied in a single decision cycle. This results in each zone to be either considered "Full" (F) or "Not Full" (NF). The state of an agent in this environment is a combination of "F" or "NF" for each of the movements which the agent controls.

This simple configuration contains four control zones: North, South, East, and West as in figure 3.1. With each zone either full or empty for these n_z zones, the number of states, n_s is:

$$n_s = 2^{n_z} \tag{3.1}$$

For the simple case, the number of zones is 4, making the total number of states 8. Extending this algorithm to a more traditional intersection shown in figure 3.3 under 8-phase dual ring-barrier signal control requires 8 zones, thus 256 states.

3.2 Action Definition

The simplicity of the intersection model implemented in this work results in each agent selecting between two actions. The actions are to either service Northbound and Southbound through traffic or service Eastbound and Westbound through traffic. If the same action is selected multiple times consecutively, this simply extends the existing phase duration.

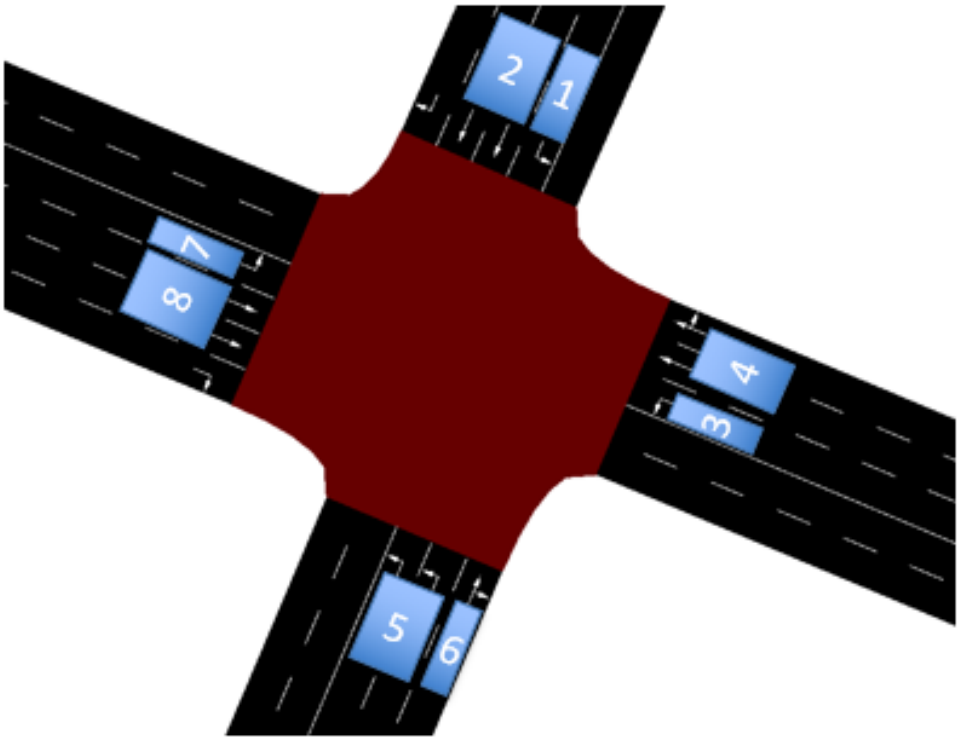


Figure 3.3: Traditional intersection with a typical lane configuration

3.3 Transition Probabilities

In order for a Markovian model to be valid, an agent's current state must be sufficient for determining the likelihood of transitioning to another state in the next time instant. Satisfying this assumption presents a significant challenge for any real-world applications for Markovian solution technique. The environment is often non-stationary which results in an agent's past states significantly altering the current transition probabilities. Additionally, multi-agent systems typically violate this assumption when cast as MDPs. In these cases, the previous states and actions of each agent in the environment impact the transition probabilities of surrounding agents.

To satisfy the stationary environmental requirement for a single traffic signal, the signal-control agent is assumed to operate a segment of roadway which is short enough such that every vehicle present in that section may be serviced in a single decision cycle. To overcome the effect of the action of upstream and downstream agents, it is established that intermediary zones, as shown in figure 3.2, are not controlled and always allow vehicles to pass through. The decisions from each agent are propagated up and downstream to other agents along these intermediaries, creating a network of linked agent-controlled zones with a known number of intermediary zones in between. Each agent is allowed to communicate both action probabilities, $P_A = \{P_1, P_2\}$, and state probabilities, $P_S = \{P_F, P_E\}$, to their neighbors on the other side of the intermediary zones as in Eqn [2.3] and Eqn [2.4]. The Q value for each action, serving North-South traffic and serving East-West traffic, is combined across all possible states. These total action utility values are converted into a probability using the softmax equation.

A key piece of developing the negotiations strategy presented in this work is converting the state and action probability values for neighboring intersections into transition probabilities. Consider a single agent, C, which exercises control over two conflicting movements, m_1 and m_2. The upstream intermediary node along the direction of travel for m_1 will be U_1 and the downstream intermediary will be D_1 as in figure 3.4. The nodes in the direction of m_2 are respectively named. In this case, the action space is $A_C = A_U = A_D = \{1, 2\}$ in reference to serving either direction 1 or 2, either along the direction of m_1 or m_2, respectively.

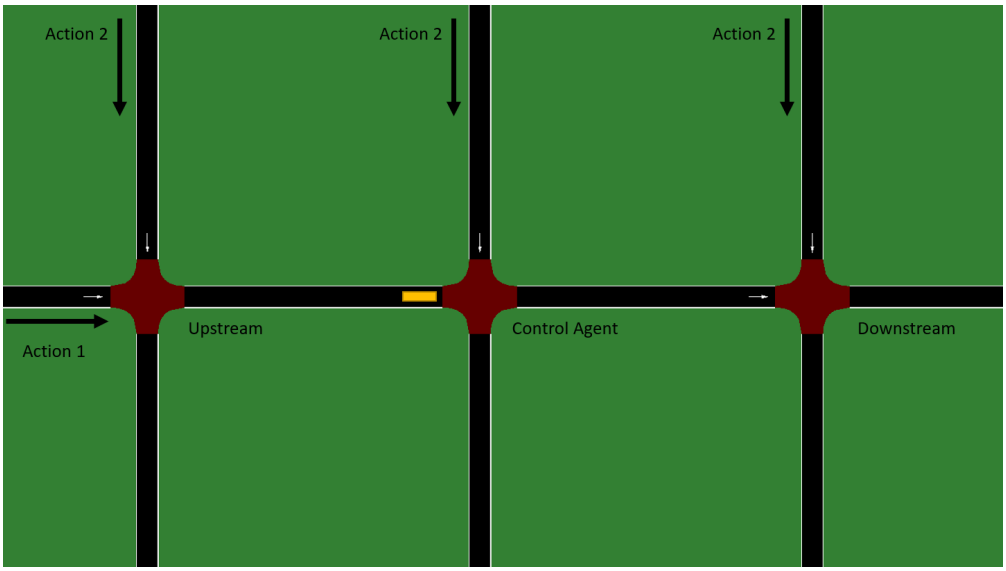


Figure 3.4: Representation of negotiating intersections

$$
\begin{aligned}
&\Pr[s, a, s']_c = \Pr(c(s') = \{F, F\} | c(a) = 1, c(s) = \{F, F\}) \\
&= [\Pr(U_1(s) = F) * \Pr(U_1(a) = 1) \\
&+ \Pr(D_1(s) = F) * \Pr(D_1(a) = 2)] \\
&- [\Pr(U_1(s) = F) * \Pr(U_1(a) = 1) \\
&* \Pr(D_1(s) = F) * \Pr(D_1(a) = 2)]
\end{aligned} \tag{3.2}
$$

In the case of this example, the transition probabilities are calculated by Eqn.3.2. This details the probability of the resultant state becoming $C(s) = \{FF\}$ when the action selected by C is "serve movement 1". This probability contains both the action likelihoods and a combination of up and downstream state probabilities. The values for state and action probability are maintained by each intersection and broadcast to neighboring intersections across the intermediary control zones during each decision cycle.

3.4 Rewards

For this work delay was selected as the sole provider of rewards. Throughput was also initially investigated. To make this decision, some measurable performance metrics for traffic intersections were considered: fuel economy, delay, throughput, and emissions. Delay both positively impacted all metrics and was more conducive to a model-based RL application.

The rewards received by the agent for each decision cycle were based on measured arrival rate for the network, A, the cycle time, C, and the number of vehicles n_v present in each control zone. For every zone, its contribution,r, to the total reward, R, was calculated according to its current state, s as:

$$
\begin{cases}
s = F & r = n_v + A * C \\
s = NF & r = n_v
\end{cases} \tag{3.3}
$$

The reward given to the intersection is the sum of all zone contributions. This reward is more readily measured using existing infrastructure such as induction loop detectors but is

capable of leveraging additional range or accuracy given by cameras or other sensors. In each case, the control zone size may be tailored to match available hardware and sensing devices.

3.5 Experimental Design

The measurable success metrics which were selected for this study were: computational time, vehicle time in network, throughput, and emissions. Specific tests were carried out in order to evaluate the efficacy of the proposed solution with regards to each metric. Additionally, the model-based negotiations were isolated in order to evaluate whether the model constructed from the dissemination of the propensity values encouraged intersections to operate more cooperatively. The presented method was also evaluated for robustness.

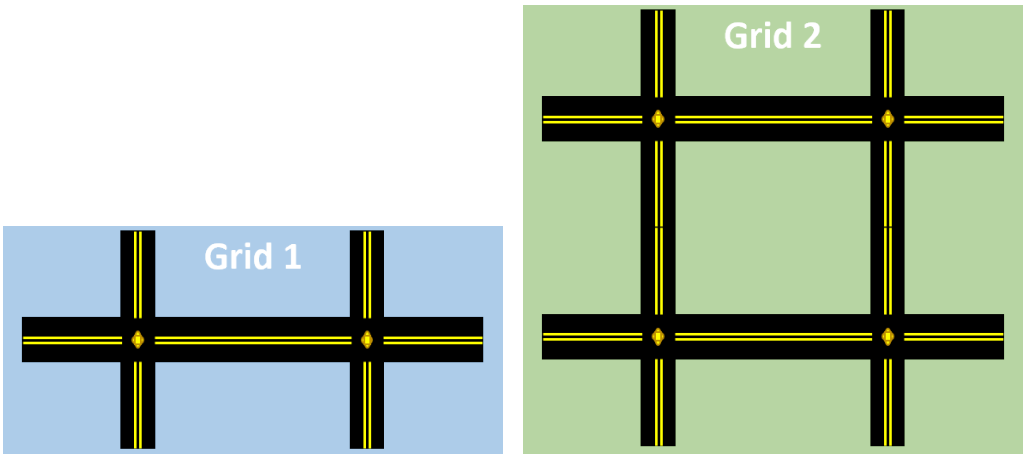
3.5.1 MATLAB Simulation Testbed

First, configurations of signals in linear fashion (figure 3.5a) and a small grid (figure 3.5b) were tested in order to ascertain the reduction of computational load by distributing the machine learning solution across all agents. In these testbeds, the vehicles only traveled in a single direction as indicated in the figure.

These tests were performed using a cellular road and traffic simulation entirely designed and constructed in MATLAB. For these tests, the measures of success are focused on computation time. Thus, the MATLAB-based cellular automata is not calibrated. In each of these tests, the presented algorithm is employed to create a solution for each signal. As a point of comparison, the same intersection arrangement is configured as a single RL agent and the system is solved using Q learning. Solving the system with an equivalent method isolates any reduction in computational time to the impact of the algorithm distribution.

3.5.2 SUMO Testbed

To test the delay, throughput, and emissions performance of the proposed solution in a more realistic scenario, a simulation for vehicle movements through a series of intersections



(a) Linear testbed **(b)** Grid testbed

Figure 3.5: Small Testbeds for Computation Time Comparison With Travel Direction Indicated

was developed in the program *Simulation of Urban Mobility* (SUMO) ([48]). Traffic flow is governed through pre-set values in the SUMO environment. For the purpose of robust design, instead of a uniform distribution of vehicles, the flows were programmed with a probability ratio of dispersal derived from Poisson's distribution. Tests for performance were carried out in this environment with the assistance of 'traci4matlab' which was an interface developed by Acosta to allow information and commands to be shared between MATLAB and SUMO ([10]). This program allowed for information to be collected in real time from the SUMO simulation to be processed by the algorithm in MATLAB and adjust the traffic signal phases to reflect the decisions made by the algorithm. A simplified network of multiple intersections in a grid format was modeled based on a network in Springfield, (figure 3.6 & figure 3.7), detailed in [54]. The constructed SUMO model was simplified to restrict turning but uses the underlying, calibrated vehicle following models in order to simulate traffic flow through the network. The network is capable of reproducing identical sets of randomly calculated arrivals using the SUMO route configurations. This allows for a fair comparison between each strategy employed.

The presented algorithm was first compared to a traditional actuated traffic controller which uses local sensors at each traffic signal in order to modify the signal cycle based on current traffic conditions. Each of these actuated controllers are able to immediately stop servicing an empty queue after a minimum time has passed and move on in their cycle to the next phase. The maximum and minimum times were selected at 10 and 45 seconds, respectively; this includes a two-second passage timer. Tests were performed to calibrate the actuated traffic time-out. In real-world modern applications, data is leveraged to tune the signal timing plan based on the observed traffic conditions. In this case, tests of combinations of 45, 60, and 90 second time-outs were evaluated and leveraged to select the 45 second time-out values.

A less measurable goal of this work is to make each agent less locally greedy and encourage more global cooperation via propensity broadcast. In order to evaluate the impact of negotiations, the presented algorithm was compared to a basic non-negotiating Q learning method. The same reward function based entirely on the reduction of delay at each agent was supplied to each strategy, non-communicating Q learning and the proposed technique. In this

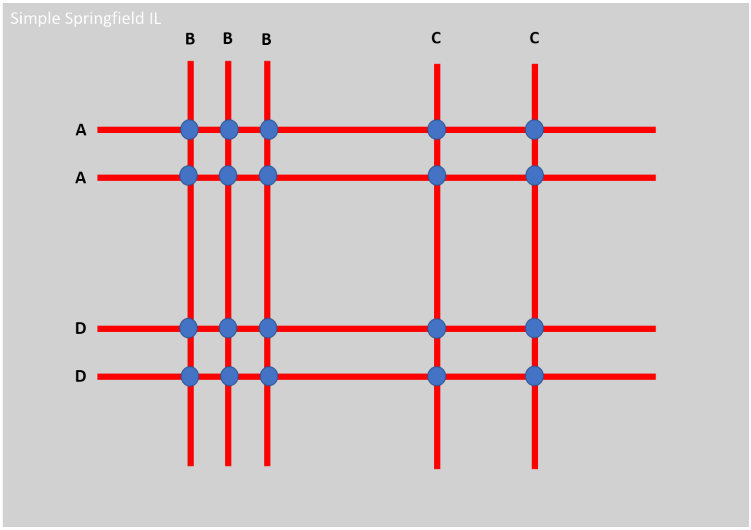


Figure 3.6: SUMO representation of testbed environment

Figure 3.7: Main testbed for examining negotiations performance

way, the only difference between the two tests is the data exchanged between intersections. In the case of the Q learning solution, the agents should individually take the actions most appealing to them.

3.6 Results and Discussion

The results in this work were divided into three categories:

1. Computational Performance
2. Performance of Negotiations
3. Traffic Management Performance

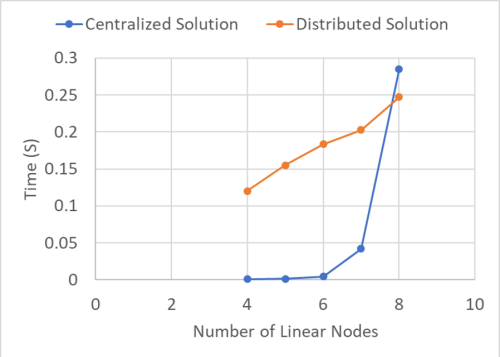
The small-scale cellular testbed constructed in MATLAB provided a proving ground for evaluating computational performance under repeatable conditions against centralized solutions. Afterwords, a larger scale, more realistic simulation constructed in SUMO allowed the analysis of traffic measures. For gauging the algorithm's impact on traffic, throughput, travel time, and emissions were selected as the chief metrics of success.

3.6.1 Computation Time Analysis

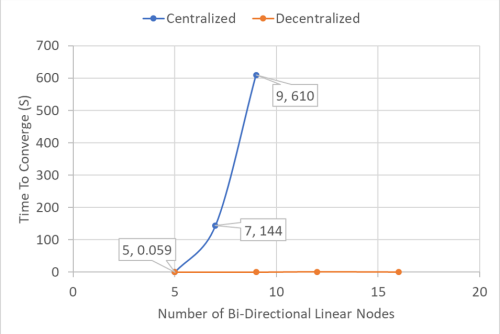
Reducing computational complexity is a key issue in any multi-agent coordination scheme. To evaluate the ability of the presented algorithm to cope with increasing complexity, the solution was employed on two testbeds (figure 3.5). Each of these testbeds were built in MATLAB using a custom-built cellular automata to allow for the easiest comparison between different machine learning techniques.

Linear Testbed (Grid 1)

For up to eight control nodes, 50 trials consisting of 120 decision cycles of uniform arrival rates were carried out on the linear testbed, Grid 1 (figure 3.5a). The resulting average computation time for a centralized and the presented methods are displayed (figure 3.8a & figure 3.8b).



(a) Unidirectional Node Convergence



(b) Bi-directional Node Convergence

Figure 3.8: Linear Testbed Computation Time Results

The time to resolve the centralized solution increases exponentially with the number of nodes. This is consistent with the rapid increase in state-action pairs which must be evaluated by the centralized method. This exponential growth in complexity renders this method infeasible for even small numbers of connected traffic signals as the complexity of each signal is increased to address multi-directional traffic flow and all permissible signal phases. This highlights the shortcomings of any solution incorporating a central controller. In figure 3.8a, each roadway in the linear network of intersections will initially only permit vehicle movement in one direction i.e. all east-west oriented roadways only serviced eastbound traffic. The second set of results indicate the convergence time when traffic is allowed to move in two directions (i.e. both eastbound and westbound traffic) are present (figure 3.8b).With a convergence time of $610s$ ($\approx 10min$), the centralized method can only reach a decision regarding the appropriate policy after the information is no longer relevant to the current traffic situation. All of these tests were performed on a conventional Intel i7 CPU.

In contrast, the convergence time for distributed methods is linear, rather than exponential with the number of coordinating nodes. This is due to the linear increase in state-action pairs as a the number of participating intersections increases.

For the centralized solution, the values in figure 3.8b are only able to extend to 8 control nodes as compared to 16 in figure 3.8a. Due to the exponential increase in computation time for centralized solutions, no further assessments could be reasonably performed for control node configurations that utilized nine or more intersections. It may be noted that when performing a 120 decision-cycle test, the simulations would require approximately 14 to 17 hours to complete with an average convergence time of approximately 600 seconds (10 minutes). Achieving performance benefits with the dstributed method would allow for more recent data to infleunce the traffic signal decisions, enabling faster reactions to unexpected conditions such as road blockage.

Grid Testbed (Grid 2)

Tests were also carried out on grid-based systems with different configurations of nodes (figure 3.5b). Each test consisted of 500 seconds of traffic with a change in traffic flow applied at 120 seconds, shifting the vehicle arrival rates to simulate a sudden change in the

traffic pattern. The result of disturbances on negotiating convergence times were observed from the experiments.

In this experiment, the probabilities of taking some action, calculated from the Q-values, converge in seven decision cycles initially. This corresponds to a 70-second time window. Following the disturbance, the probabilities, take around 150 seconds to converge to a new steady-state (figure 3.9). In this figure, each of the colored lines represents the probability of taking action 1 (i.e. "service north-south") for a separate intersection.

In each iteration, the estimates for upstream and downstream actions are held over from the previous time instance. Initially, this allows for faster convergence. However, when perturbed, a significant increase in convergence time is observed. Agent's Q-values or other learned values are randomly reset, dropped out, or otherwise re-initialized when environments or other factors change. To perform this sort of reset in a traffic signal, it would be necessary to develop a suite of criteria describing when a significant enough change has occurred. Alternatively, the values for probability could be dropped between each set of negotiations. The selection of methodology should be driven by the performance gains in an environment with more realistic traffic conditions.

At the point of disturbance, the convergence time for each intersection increases significantly. The source nodes are informed of the new arrival rates (i.e. the south-most nodes) and given the new rate of northbound vehicle arrivals. The Q-values from the new arrival values contradicted those from the old traffic pattern. The shift in predicted state-action value should be the sole contributor to this increase in negotiations time. After five decision cycles, the negotiations time returns to normal.

From the experiments, several different state and action probability distributions were observed, but the system reached stability when given several decision cycles to settle. The mechanisms by which these action probabilities converge to a set of values are beyond the scope of this work, however some observations were made during testing. In every case, the values for action and state propensities showed strong convergence towards an optimal set of values, except in the case of a very specific set of conditions in which the demand levels and current distribution of vehicles were mirrored, causing the system to oscillate between two sets of propensities. To avoid an infinite loop due to this oscillation, one of the two

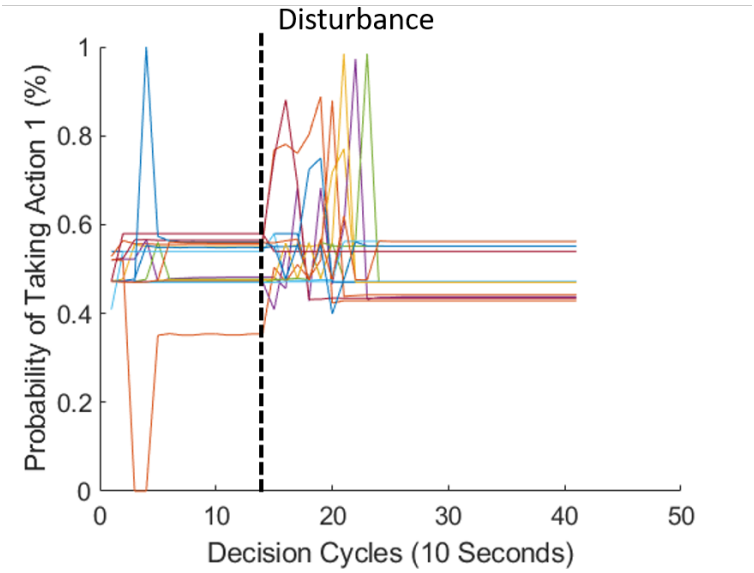


Figure 3.9: Probabilities of Node Actions Stabilizing

propensities was selected at random. Understanding the underlying process by which these values are reached without any sort of learning rate or decay should be a target for future work as the impact on system performance can be significant.

3.6.2 Effect of Pseudocells on Convergence

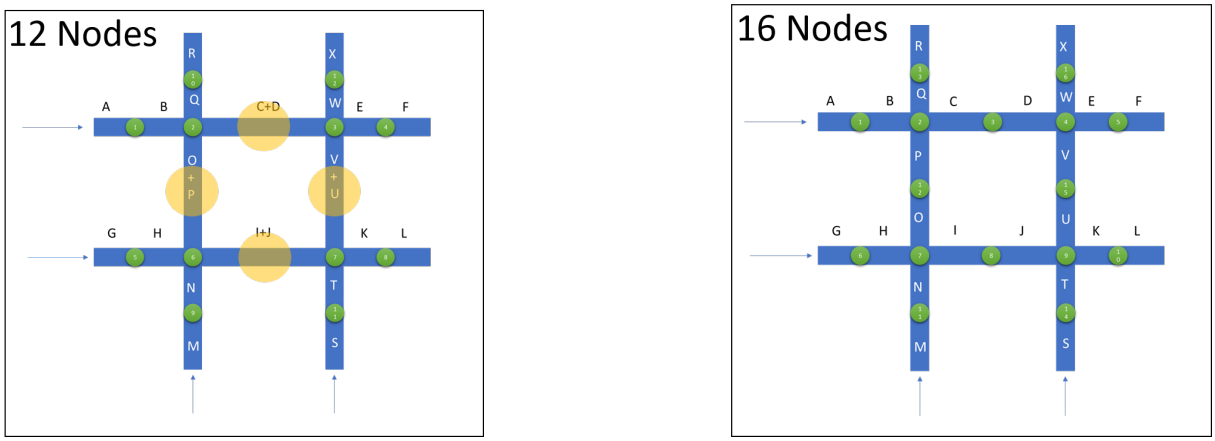
The impact of system configuration was explored briefly during the small-scale grid experiments. The difference between a 12 node and 16 node configuration was of particular interest. (figure 3.10).

An intersection was simulated along the roadway between two major intersections. This addition was made in order to estimate the performance of the larger-scale configuration of intermediary cells described in figure 3.2 which leveraged sets of artificial divisions along a roadway to deal with long distances between signalized intersections. The absence of these artificial intersections is pointed out with yellow highlighting in figure 3.10a. This addition reduced the computation for the 16-node configuration such that it outperformed the 12-node configuration in terms of convergence time (figure 3.11). This implies that the system configuration has a significant impact on the performance of the algorithm.

Specifically, the addition of a simulated "trivial" node in between two major intersections enabled neighboring nodes to more quickly reach a decision. Recall that these intermediaries are "uncontrolled" lengths of roadway. It follows that these nodes are designed such that no conflict exists, and they would always reach the decision to "serve" the only movement they control. For example, if the artificial zone existed along an east-west roadway, it could only choose action 2, serve east-west. This set of known decisions appears to have a positive impact on the length of time taken for the negotiations strategy to converge. Future works will further investigate the impact of these artificial nodes on the performance of the network.

3.6.3 Comparison to RL baselines

Although small-scale testing on simple systems provides useful environments to ascertain the impact on computational complexity, any performance measures should be collected in an environment with a more realistic model for vehicle traffic. In this case, SUMO was leveraged



(a) 12 Node Configuraiton **(b)** 16 Node Configuration

Figure 3.10: Grid Format Testbeds

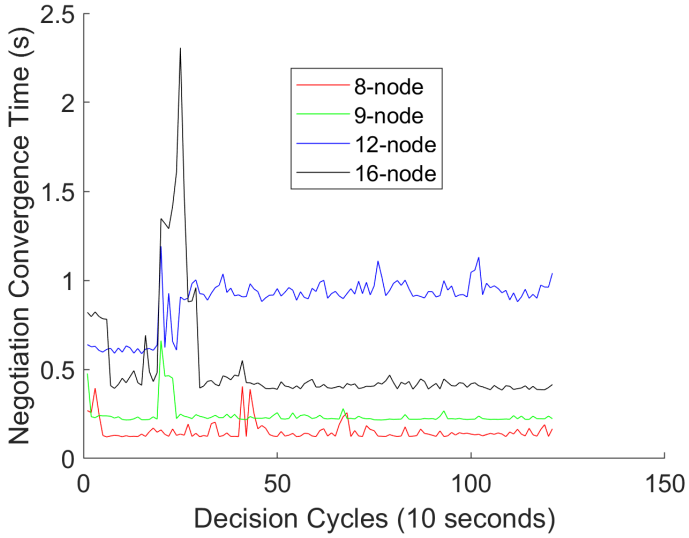


Figure 3.11: Effect of Demand Shift on System Convergence Time

to produce a larger-scale simulation. The underlying mechanics of SUMO, specifically the continuous-time vehicle following behavior, enable a more real-world look at the performance of the negotiations algorithm which makes observations and decisions in 15 second intervals.

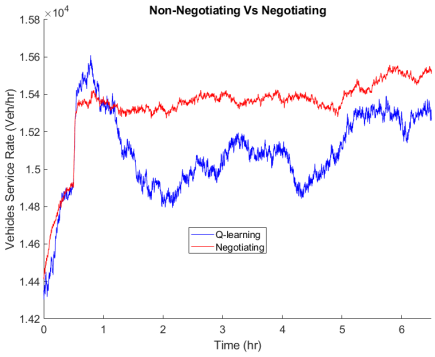
In order to isolate the impact of Negotiations, a set of five simulations were carried out containing data for seven hours of traffic for each of the Negotiating, Non-negotiating basic Q learning, and Dyna-Q learning algorithms. A total of $6.5hrs$ were included in the analysis to cut off the last portion of the simulation in which traffic was no longer being supplied to the simulation. Each of the figures in this section contain the reduced data from the five negotiating algorithm tests and the five non-negotiating "baseline" tests. This data was evaluated using a $1hr$ moving-window average to smooth the performance. It can be noted that, based on the standard deviation in table 3.2, this smoothing primarily aids the non-negotiation Q learning and Dyna-Q learning as the performance of these two strategies was much more varied.

Comparison to Q learning

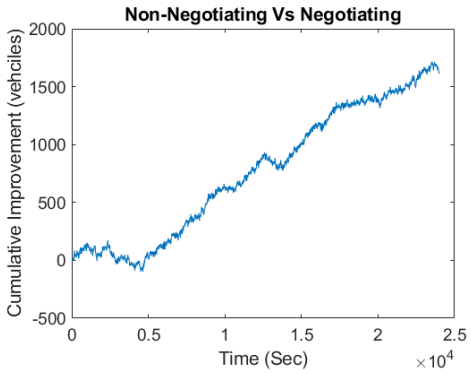
Based on the data noted in ([54]), static arrival rates of $1440veh/l/h$ east-west and $360veh/l/h$ north-south was provided to the grid of traffic signals. In the comparison to a baseline Q learning algorithm, static arrival rates were tested for the duration of the 7 hour simulation. This should enable the baseline RL implementation sufficient time to fully explore the state-action space and arrive at a solution for any equilibrium traffic conditions which emerge after the intersection network is saturated.

The throughput-per-hour difference between non-negotiating Q learning and the negotiations scheme were calculated using a rolling window in figure 3.12a. For the majority of the simulation, the average service rate for the negotiations algorithm exceeds that of the non-negotiating strategy. This results in a net positive trend in the cumulative service (figure 3.15b).

There is brief period during the first hour of the simulation for which the difference in average service rate declines (figure 3.12a). The intersection network in-simulation initially contains no vehicles, and SUMO is allowed to populate the environment through the entry-point nodes along the perimeter of the network. During the first hour of each simulation



(a) Throughput Per Hour



(b) Cumulative one-hour-average difference in throughput

Figure 3.12: Cumulative difference in throughput

presented in this work, the control strategies are dealing with dynamic conditions as the intersections and connecting roads fill with traffic until a steady-state is reached. Prior to steady-state, the distribution of vehicles in the network may be sparse.

The presented method assumes that each intersections' actions impact neighboring signals, and these agents are provided with an estimate of incoming vehicles based on the arrival rates at the network perimeter. However, while the cars in the network are sparsely distributed, both the assumption of impact and the estimates may not hold true. This causes the presented negotiations method to incorrectly assess the value of each potential action. In these cases, the baseline Q learning and it's locally greedy actions outperform the negotiations.

After enough vehicles arrive in the system to appropriately connect each intersection, the assumptions made in this work become valid, and the performance of the negotiations method improves. This behavior indicates that, during a real-world application, it may be necessary to either improve the accuracy of vehicle arrivals at individual intersections or switch to a locally greedy strategy when this value is sufficiently low.

If the total number of vehicles serviced is low enough, the impact on actions of upstream and downstream agents is not as high. In the background, the model is using vehicle arrival values set by the exterior intersections. This means that the interior nodes estimate future rewards with an estimated nonzero arrival rate. This is not the case as the rates will be zero until the upstream intersections allow vehicles to pass. This could result in sub-optimal performance which would be exacerbated in low-traffic conditions. However, when the intersection network is sufficiently populated with vehicles, these estimates are good enough and the negotiations approach average throughput passes that of distributed Q learning within 10 minutes. The improvement in throughput peaks at around 4.32 hours when the negotiations algorithm succeeds in servicing, on an average, $539 veh/hr$ more vehicles compared to the "non-negotiating" Q learning method. In the experiments performed, the estimates for vehicle arrivals which each intersection relied on to calculate rewards were kept static throughout the simulation. After the actual arrival rate shifted, these values remained fixed to the original arrival rates at the edge nodes of the traffic network. For a better real-time estimate of arrival rates, each intersection could be configured to record average arrivals

over a short window. Much like the 'advice' parameter in ([15]), these arrival estimates could improve the performance during this period.

Analysis of the emissions provides further insights into these patterns. The delay at each intersection directly impacts the emissions since SUMO uses vehicle speed and stoppage to estimate this value. Negotiations consistently result in a reduction in emissions shown in figure 3.13.

Since the traffic signals are the principle source of delay in the system, the total time spent by any one vehicle in the system should be a good measure for how much delay reduction each strategy provides. In this case, the summary statistics in table 3.2 indicate that the non-Negotiating strategy performs worse than both the negotiating strategy and actuated control strategies. This is reflected when evaluating a 1-hour windowed average of vehicle time in system over the course of the simulation in figure 3.14.

In this case, the values indicated that vehicles consistently spend more time in the system when the non-negotiating control strategy is employed. The values towards the beginning of the simulation reflect a relatively empty network in which locally greedy strategies may have equivalent positive results to a more cooperative controls scheme. However, as the network becomes more congested, the time savings due to the Negotiations strategy reach a peak of $20.9sec$ at $4.6hrs$ into the simulation. The global average was a $16.91sec$ savings, and between the $2hr$ and $5hr$ time steps, the time savings never fall below $16.2sec$.

The results indicate that the broadcast action and state probabilities provide better performance at the network level which was the goal of this work. In practical applications, the static arrival estimates could be set for times of day, events etc. or these values could be directly observed from the environment. While more study is required to determine whether this change would produce the desired improvement in long-term performance, the results presented here indicate that even supplying static arrival values based on prior information would be sufficient to produce an improvement in performance using the negotiating strategy individual agent Q learning.

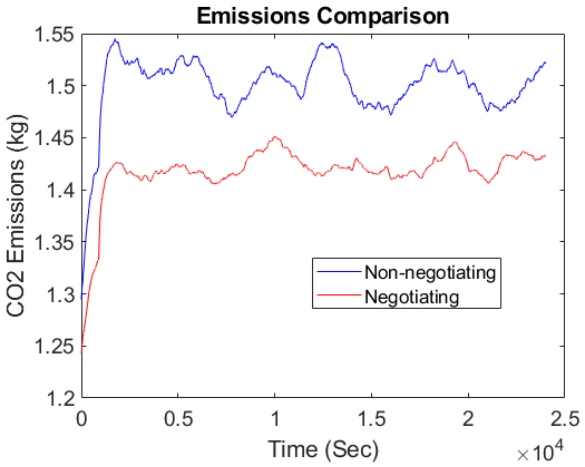


Figure 3.13: One hour emissions rate difference between Negotiating and Non-negotiating

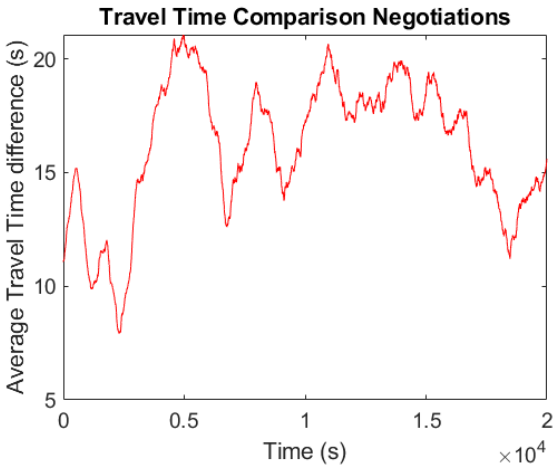


Figure 3.14: Hour-window Average Travel Time Comparison Non-negotiating - Negotiating

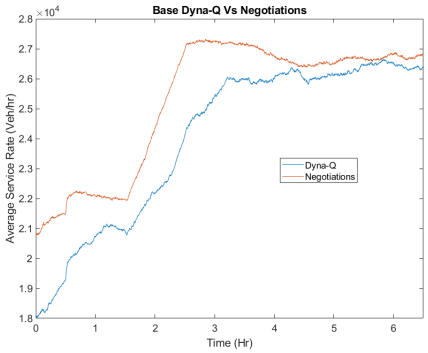
Comparison to Dyna-Q Learning

A second set of tests was carried out to compare the agents to Dyna-Q learning. This extension to base Q learning leverages an underlying model which samples from past visited states and actions in order to more quickly discover an optimal solution. In changing environments this algorithm struggles as, if the states are not visited initially, the agent may miss shifts in reward simply by arriving an an optimal solution early on and not fully exploring the environment.

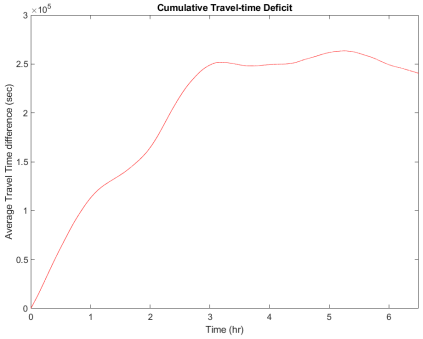
In this case, the arrival rates rates start from $1440veh/l/h$ along eastbound and westbound lanes and $360veh/l/h$ along all northbound and southbound roadways. At $t = 2hr$, the values change to $1080veh/l/h$ along the north-south corridor and $720veh/l/h$ along the east-west roadways. The negotiations algorithm consistently outperforms the Dyna-Q structure. From $t = 0$, the benefit of the underlying model is $2740veh/hr$ (figure 3.15a). As the arrival rates shift, the number of vehicles arriving increases to match the additional influx of vehicles. The negotiations strategy responds faster to this increase. While the exact mechanism is not known, a reasonable assumption is that the ability of the negotations-based algorithm to update and simulate any state irrespective of visitation aids in such situations. In the case of delay, the aggregated delay across all vehicles over the course of the simulation reaches $73.2veh - hrs$ with the maximum difference per-vehicle of $146sec$. After $5hrs$ in the simulation, the density of vehicles in the network is such that both algorithms are made equivalent. Whether this is entirely due to vehicle saturation or a difference in performance was not fully explored. It is likely that the Dyna-Q algorithm's underlying model becoming more accurate plays a part in this. However, the continued performance of the negotiations in terms of throughput suggests that this algorithm is superior and the delay is at a minimum for the current level of congestion.

3.6.4 Comparison to State-of-the-art baseline: Actuated Signal Control

Two additional tests to analyze the traffic management performance of the presented method were conducted using the Springfield SUMO model:



(a) Throughput Per Hour



(b) Cumulative one-hour average delay aggregation

Figure 3.15: Cumulative vehicle delay

1. A static arrival rate of $1440veh/l/h$ east-west and $360veh/l/h$ north-south

2. Shifting arrivals at $2hr$ to $720veh/l/h$ east-west and $1080veh/l/h$ north-south

Both of these tests compared the presented method's performance against a more traditional actuated traffic signal strategy, with results summarized in table 3.1. The actuated control algorithm leverages the existing loop detectors at each signal to determine whether to service the queue until a pre-set maximum time has been reached or to terminate service early if no vehicles are present. The timeouts for the actuated traffic were set up as uniform for both east-west and north-south. In practice, these values are tuned to establish a coordinated "wave" for the signals along an arterial. Rather than specifically tune the values of both the actuated controllers and the negotiations strategy, two sets of values were selected for the actuated controllers, 45 and 90 seconds. The 90 second value reflects a more typical real-world value for signal time-out, while the 45 seconds is much shorter to account for the high traffic volume and smaller space. Additional combinations of 45, 60, and 90 second time-outs for East-West and North-South traffic.

While the values for throughput improve with lopsided time-out values, maximum delay is increased by a minimum of 164 seconds in the case where North-South is serviced more heavily. The chosen time-out values are representative of the performance in this regard.

Although none of these tested values are necessarily tuned to the specific configuration presented, a strength of the Negotiations algorithm is the lack of need to specifically address each unique traffic combination. Analyzing both short and long time out values provides a point of comparison for this study. In a real-world application, the initial model parameters could be tuned the same way the time out values are tuned to achieve better, more consistent performance.

In the case of actuated traffic, each seven-hour simulation is identical as SUMO arrivals are randomly initialized but carried out identically for repeatable tests. In order to provide a complete comparison to these values, another set of five simulations were carried out for the Negotiations algorithms. The same $6.5hr$ were included in the analysis as in the comparison to non-connected Q learning. Each of the figures in this section contain the reduced data from both the five negotiating algorithm tests and one of the single, identical simulations for

Table 3.1: Summary of Actuated Time-out Performance

Time-out EW:NS	Mean delay (sec)	Median delay (sec)	Max delay (sec)	Min delay (sec)
45:45	169	163	449	49
60:60	170	160	644	50
45:90	180	163	613	48
90:45	223	184	756	54
90:60	173	159	686	47
60:90	223	205	847	52

actuated signal control. This data was again evaluated using a $1hr$ moving-window average to smooth the performance.

Static Arrival Rates

In the first set of tests, static arrival rates of $1440veh/l/h$ were used for all eastbound and westbound lanes with $360veh/l/h$ along all northbound and southbound roadways. Within this test, two sets of maximum time out values were tested for the actuated signal controller:

1. 45 seconds
2. 90 seconds

The performance of these actuated signals versus the negotiations algorithm was compared across all performance metrics.

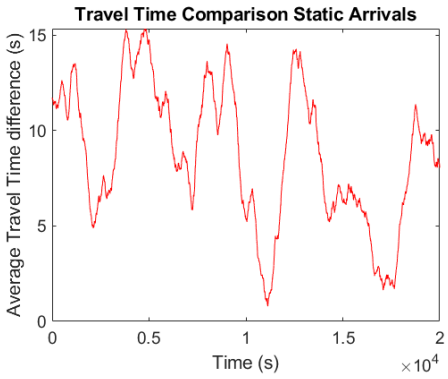
Individual vehicle travel time was the first key performance indicator analyzed. Over the recorded two hours, each vehicle entering the network was assigned a hidden identifier and timer. Once these vehicles exited the simulation, the total time spent driving was recorded. The results from each of the simulations are presented in table 3.2.

Because the only source of delay in the network are the signals, any lost time or time savings must be caused by the signal control strategy. Both the mean time in-network and the max time in-network are improved over the actuated traffic under static conditions. A one-hour windowed average was calculated to visualize the change in travel time over the course of the simulation (figure 3.16).

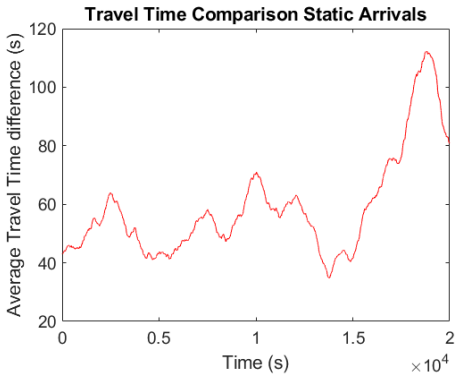
In the case of static arrival rates, the negotiating strategy always produces a lower travel time than that of the actuated signals. The difference peaks at $1.5hr$ when it takes $14.3sec$ more on average for the actuated traffic to pass through the network of signals. At $5.9hr$, the difference tails off as the windowed average captures more instantaneous values. This is not representative of the system performance which, on average, results in $9.8sec$ saved by the Negotiations algorithm. The margin of this improvement is indicated by the positive travel time difference in figure 3.16. The apparent trend towards zero difference captured by the graph is likely a side effect of the total vehicle volume being supplied to the grid. At a certain point, the vehicles will jam regardless of control strategy employed. However, the

Table 3.2: Vehicle Time in Network Summary

Static	Median	Mean	Std	Min	Max
non-Negotiating	166	174	71	47	837
Negotiating	160	160	41	50	313
Actuated	163	169	51	49	449
Changing	Median	Mean	Std	Min	Max
Actuated	190	216	94	56	576
Negotiating	213	221	63	50	589



(a) Actuated time out at 45 seconds



(b) Actuated time out at 90 seconds

Figure 3.16: One-hour rolling average travel time for actuated - negotiating strategies

Negotiations algorithm is able to cope with the high traffic volume more effectively in terms of travel time than the actuated signals for the course of the six hour simulation.

Notably, the 90 second time out actuation performs significantly worse than the 45 second timeout. This is likely a result of the increased congestion in the vehicle network favoring shorter cycles to keep traffic moving and avoid jam conditions. As the network approaches saturation, the vehicles will take longer to traverse the network irrespective of signal strategy. This results in a slow downward trend in the difference which starts after $2hr$ of continuous vehicle arrivals. Further evidence of this can be seen in the 90 second time out plot which appears to reach some jam or backup condition around $4.2hr$, causing the travel time difference to peak at over $100sec$ worse for the actuated strategy.

Emissions for the network can be analyzed in order to gain some additional insight into the delay and travel time values around the arrival rate shift. Due to SUMO's method for penalizing non-moving or slow-moving traffic, the impact of the algorithm on delay can be estimated by looking at emissions data.

In figure 3.17, a one-hour windowed average of emissions for the negotiations strategy was subtracted from the same for actuated data. In the $45sec$ time-out case, the negotiations algorithm experiences a maximum of $425kg/hr$ improvement over actuated signal control network-wide. The mean and median for this emissions data are both near $320kg/hr$ at $325.8kg/hr$ and $322.1kg/hr$ respectively with a standard deviation of only $37.11kg/hr$. The peaks in this plot should coincide with times where the heavier traffic along the east-west arterial roads are stopped to allow north-south traffic to move. For the $90sec$ timeout case, the improvement is even more dramatic as the longer time out causes heavy traffic backups in either direction, stopping vehicle motion for extended periods of time. At $4.2hr$ the emissions increase significantly, again indicating some jam or intersection blockages due to traffic buildup. Tuning or modifying the maximum phase times for the actuated signals would likely improve the performance of the the actuated signals in the static case. However, if any changes or external stimuli impact the arrival rates for the network, any tuning would be rendered inaccurate.

The third metric of focus was total vehicle throughput. The goal of such a data broadcast scheme is to improve the performance at a network level rather than simply boost the

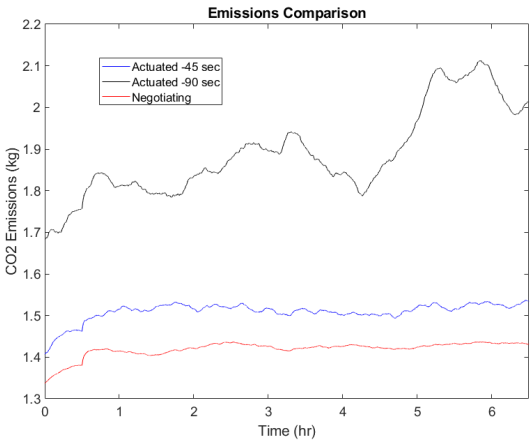


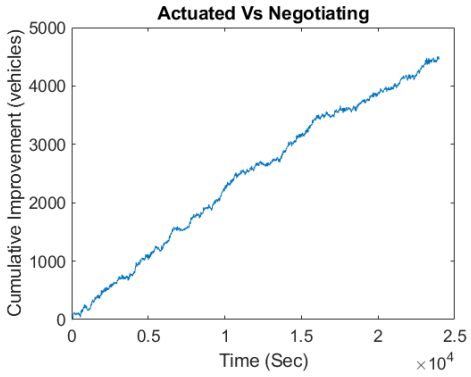
Figure 3.17: One-hour rolling average emissions created difference Actuated - Negotiating

performance of individual intersections. In this case, total throughput was measured by evaluating the number of vehicles able to exit the network at each time instant. These values were assessed with a one hour rolling average window in figure 3.18.

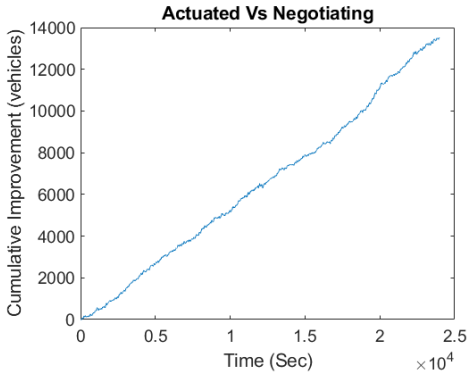
The MAL algorithm outperforms the actuated signal consistently throughout the simulation. As a result, the cumulative impact over the course of $6.5hr$ was a $4393veh$ total improvement in service. As throughput does not directly drive the RL agent rewards, it is not expected for the rate of throughput to continually improve. However, delay and throughput should be somewhat correlated as any slowdown or stoppage increases lost time. Whichever direction is more heavily congested will see a more significant increase in delay, thus encouraging the RL agent to serve that direction. At least in this simplified model, that alone leads to increased throughput. This is further reinforced by the $90sec$ time-out values. These throughput values are likely heavily influenced by the fact that there is consistent traffic for both directions due to the grid arrangement of the signal network. This causes the actuated algorithm to time out more often than not, leading to an over-servicing of the less-congested north-south corridors. This increases delay and significantly impacts the throughput achieved.

Shifting Arrival Rates

In the second set of tests, arrival rates of $1440veh/l/h$ for all eastbound and westbound lanes with and $360veh/l/h$ along all northbound and southbound roadways change at $t = 2hr$. The new values after this time are $1080veh/l/h$ along the north-south corridor and $720veh/l/h$ along the east-west roadways. This shift in arrival rates is sudden in-simulation, but may reflect real-world events such as vehicle collisions or roadway blockages which impact the flow of traffic in and around networks of signalized intersections. It is important to note that for this test the arrival rate estimates for the negotiations algorithm remained unchanged. This simulates a worst-case-scenario where static rates have been supplied similar to the tuned traffic signal timing structure of modern intersections. In a real-world implementation, the performance could be improved significantly by recording average arrival rates at the network or arterial ingest points.



(a) Actuated time out at 45 seconds



(b) Actuated time out at 90 seconds

Figure 3.18: Cumulative throughput benefit Negotiating - Actuated

Analyzing the travel time values in table 3.2 shows that the actuated signal controller with a 45 second time out does outperform the negotiations algorithm in terms of maximum and average travel times when the network undergoes a shift in arrival rates. More information is available from the plot showing $1hr$ moving averaged travel times (figure 3.19).

The travel times during the initial phase of the simulation favor the negotiations algorithm with a peak $15sec$ time savings at $1.6hr$, more or less mirroring the previous results. However, as soon as the arrivals change, the travel time shifts in favor of the actuated signal controller, closing the gap until the actuated outperforms the Negotiations at $2.2hr$ into the simulation. The RL agent is still making decisions according to the model it has learned with the initial traffic pattern. Additionally, the rewards structure directly uses a set of supplied static arrival rates to estimate the number vehicles in each control zone. This was necessary as the range of sensor equipment at signalized intersections is often limited, especially in the case of induction loop detectors. In these cases it was necessary to develop an estimate for the number of vehicles waiting in a control zone without the ability to directly measure the quantity. For this work, the static initial arrival rate was leveraged. Even with this limitation, an apparent trough is visible at about $4.8hrs$ into the simulation, $2.8hrs$ after the demand shifts. At this time, the algorithm appears to be reducing the gap.

In future works, the new arrival rates could be broadcast from network ingest nodes, edge intersections, as a 15 minute or 30 minute average arrival rate. This could allow the agents to quickly and easily update their internal models and adapt to new traffic situations much faster and without any specific tuning or prior knowledge.

Despite the detriment to travel time, the emissions rates remain in favor of the negotiations algorithm (figure 3.20b).

In this case, performance increases consistently, even becoming more pronounced at the point of the arrival shift, $2hr$. In this case, both the negotiations strategy and the actuated strategy experience an increase in overall emissions on the order of $0.7kg$ due to the increased arrival rates. However, the traffic remaining in the network at the point of the shift is distributed according to the initial east-west heavy flow. The negotiations algorithm continues to service these vehicles according to the previously learned strategy, resulting in an immediate benefit to emissions. The later dip in performance at $4.3hrs$ could be explained

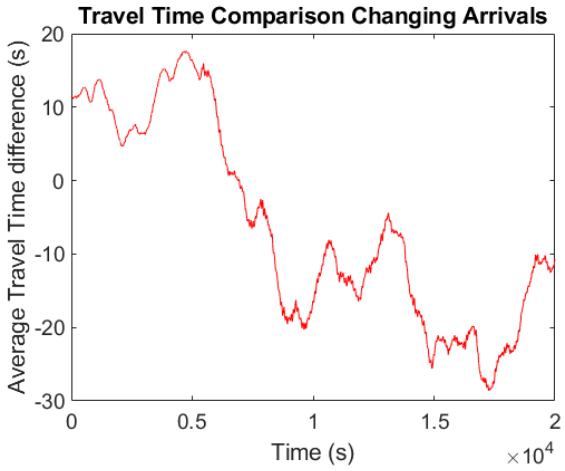
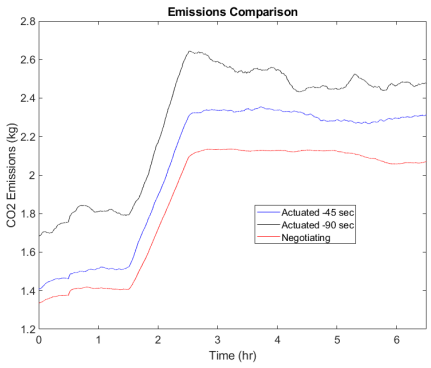
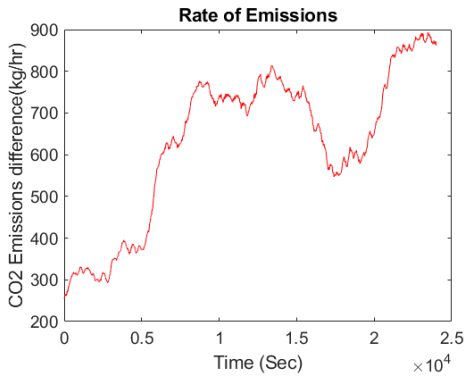


Figure 3.19: Hour-window Average Travel Time Comparison Actuated - Negotiating



(a) Cumulative Emissions With Changing Arrivals



(b) Rate of Emissions With Changing Arrivals

Figure 3.20: Hour-window Average Emissions Comparison with Changing Arrivals

by the gradual shift in vehicle distribution due to the more north-south heavy flow. In this case, the negative impact on the Negotiations algorithm was offset by its ability to clear out any built-up traffic along the east-west corridor.

Since the rewards for the presented algorithm were purely delay-based, the resulting improvements in throughput were all side-effects of maintaining traffic movement through the vehicle network. When the arrival rate shifted mid-simulation, the negotiations algorithm again outperformed the actuated signal control with respect to total number of vehicles serviced. Visually, there was a negative trend in the difference in throughput rate for the delay-based rewards (figure 3.21). As an additional experiment, a weighted rewards structure was implemented. With this, 20% of the rewards were supplied by directly measuring the number of vehicles served in a given cycle. The other 80% were calculated according to the delay-based rewards structure. These results are shown as the orange line in figure 3.21. The initial poor performance of this rewards structure is due to the lack of any prior model or service rate estimate for the network. As the agent makes observations, at $10000sec$, the performance for the partially throughput-based rewards structure surpasses that of the delay-based rewards (figure 3.21).

This experiment opened an exciting array of possibilities indicating that this strategy is compatible with combined reward functions consisting of multiple driving factors. The negotiations core strategy was designed to be rewards-agnostic, operating only on the reported utilities. However, the exchanges and data presented in this work were specifically adapted to the delay-based rewards structure. The results from this test indicate that the overall premise is indeed transportable to other rewards frameworks although the initial performance suffers without a tailored initialization function or initial set of observations.

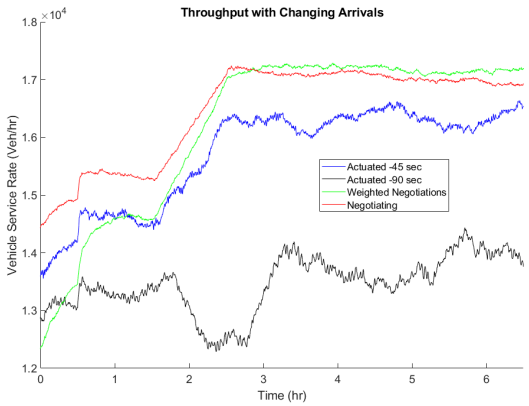


Figure 3.21: Hour-window Average Throughput Rate Comparison Actuated - Negotiating

Chapter 4

Algorithm for Automated Platoon Formation

Many current offerings in the autonomous driving space include onboard AI and special hardware in the self-driving vehicles [aut, int, nvi, pix]. This creates an inherently distributed computational environment where each vehicle may be expected to make its own decisions, impacting those around it. The goal of adapting a Negotations-based agent as a supervisory controller in this environment is to enable agents to more accurately assess the long term benefits of changing the current driving behavior. This can lead to ad-hoc platoon formation or dissolution in collaboration with other agents driving along the same stretch of roadway. Ideally, such a supervisory controller would allow agents to more intelligently take actions to maximize some underlying cost function rather than adhering to a set policy. To cast the platoon-formation problem as state-based (MARL) problem, each CAV, C^i, is assigned two separate machine learning agents:

1. A forward-facing agent, C_f^i
2. A rearward-facing agent, C_r^i

Each of these agents possesses the 5-tuple state, action, reward, transition probabilities, and discount factor, S, A, R, T, γ, which defines the agents' behaviors. Each forward facing agent, C_f^i, negotiates with the rearward facing agent, C_r^j, where agent C^j is the CAV immediately leading C^i. Specifically, each agent communicates probabilities for:

1. Adopting a potential speed regime

2. Being interrupted by a non-connected vehicle

The goal of these negotiations is to enable leading and following CAVs to cooperatively determine the best speed regime in response to these likelihoods. The selected speed regimes will either result in the agents forming a platoon or not. An agent choosing not to platoon may either move apart, or remain at the roughly the same distance. If two vehicles are within $50m$ of one another, we consider them to be platooned. The reward metrics for each agent include its immediate fuel consumption, and the difference between desired and current speed. A long-term benefit metric which could be calibrated or set based on the projected route was also tested as a modification. This optional metric is meant to enable agents to gauge estimated total benefit for platooning based on the maximum time that the vehicle would remain a part of the platoon. Each reward element is normalized and combined with a weighted sum. This technique is common and can be seen in other works to incorporate multiple reward sources often derived from features or signals of differing units [**? ?**].

4.1 CAV Agents

Consider three vehicles, X, Y, and Z. The forward-most vehicle, "X", is the leading vehicle. Immediately behind "X" is "Y", which is followed by "Z" in the trailing position as in figure 4.1. The center vehicle, "Y", will serve as an example to explain the proposed 2-agent per vehicle method.

4.1.1 Forward-Facing Agent

First, consider Y_f which will evaluate its vehicle, Y, and leading vehicle, X. In order for CAV Y to evaluate platooning with leading CAV, X, a "game board" is established for the forward-facing agent Y_f and vehicle X. This game board, $\beta_{(Y,X)}$, illustrated in figure 4.1, consists of 10 uniform 100m tiles along which vehicle Y will move towards vehicle X. In each game, the trailing vehicle's forward-facing agent, Y_f, is initialized in one of these grids based on its relative location to the leading vehicle, X. The total state space for agent

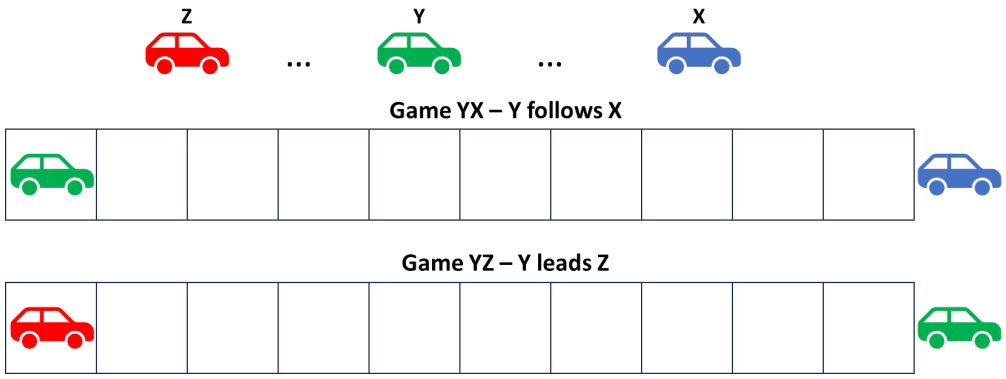


Figure 4.1: Example games played by vehicle Y: YX and ZY

Y_f, is defined by combinations of agent Y_f's speed, S_{Y_v}, in miles per hour, and position, S_{Y_g}, in grids relative to X. To prevent state explosion, the speeds for the agent are defined as 10mph regimes from $50mph$ to $70mph$. This could be reasonably expanded to fit most typical highway speed ranges. It would also be reasonable to redefine size of the grids or set a maximum distance between negotiating vehicles.

One other assumption in many discrete Markovian machine learning solutions is the assumption that, given sufficient time, all states will be visited enough such that the values and transition probabilities between them is well known. From that information, the resulting policy can be considered optimal. In the case of vehicle platooning, road conditions, traffic, unexpected events, etc. can cause the transition probabilities between states to change or shift. To be more effective, two methods for offline learning or simulated experience were explored. In these instances, models of the environment were created and stored locally with each agent. Between real observations and update steps, this model was refined and leveraged to give agents a method for bootstrapping experience, even in previously unexplored states.

The action set for agent Y_f is also equal to the set of speeds in the state definition:

$$A_{Y_f} = S_{Y_v} \tag{4.1}$$

This state and action set establishes the environment and actions for a "following game" which agent Y_f will play.

Game Board

In order to ensure that the system adheres to the properties and assumptions of a traditional Markovian environment, the board is defined relative to the speed of vehicle X.

$$v_{\beta_{(Y,X)}} = v_X \tag{4.2}$$

By defining the board this way, agent Y_f is the only mobile actor in the environment $\beta_{(Y,X)}$. Vehicle Y and associated front-facing agent Y_f move along the surface of the board, traversing

a number of tiles equal to the difference in relative speed between Y and X given a fixed time duration, ΔT. For this work, ΔT was set consistently to $15sec$. As the vehicles draw closer, if they are considered to be platooned, a traditional method for instantaneous speed control should take over and provide optimal, stable following behaviors. The $15sec$ is only considered acceptable for the non-safety-critical supervisory speed regime control exercised by the Negotations algorithm.

Offline Learning

An underlying transition probability set, T, enables Y_f to perform offline learning for this game. This model could be learned over time directly to predict transitions to each state, a combination of speed and distance between vehicles X and Y. However, sufficient state visitation may only produce a sparse transition matrix using this method. Instead, the problem of state prediction was decomposed into speed and distance estimation. The sources of uncertainty in this estimate are interruptions from surrounding, non-connected vehicles, and the probibalistic actions of the ather CAVs in the environment.Two techniques were explored to model the predicted distances between each vehicle.

In the first method, all agents locally store a more finely discretized representation of the road surface (figure 4.2), with $1m$ resolution, along with local measurements for traffic density. This finer model, in addition to the action likelihoods are to used to perform sets of Monte-Carlo simulations for the upcoming decision cycles.

The locally stored roadway models will be referred to as the "translation layer", ϑ_V, associated with an individual vehicle V. The translation layers consist of one-foot-length grids which are defined as moving with the local average traffic speed.

$$v_{\vartheta_V} = \overline{v_{traffic}} \tag{4.3}$$

The likelihood that a CAV is interrupted by a non-connected vehicle in any grid in the translation layer is then:

$$Pr_{interrupt} = \overline{\rho}_{traffic} * \frac{mile}{5280ft} \tag{4.4}$$

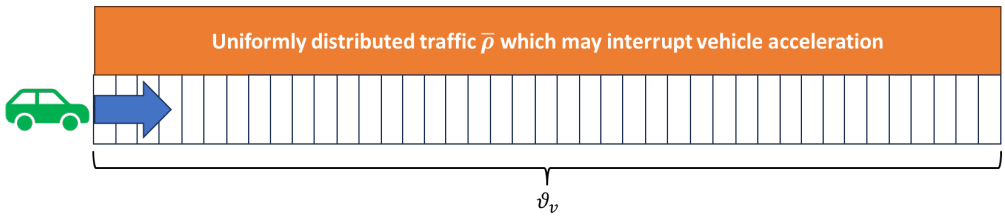


Figure 4.2: Densely discretized translation layer with uniform traffic source assumed in adjacent lane.

where the expectation for the presence of a vehicle is given by the sensed average surrounding traffic density, $\overline{\rho}_{vehicles}$ recorded in vehicles per mile.

In the second method, a linear regression was performed to find a relationship between the difference in chosen speed for a leading and following agent-pair and the resulting distance after a single decision cycle. This regression was performed for a set of traffic densities, generating a surface. Based on the local traffic density and selected speeds, this model directly produced the estimated end distance from the linear fit.

In both methods, agent Y_f relies on information from X_r, the rear-facing agent contained on CAV X in order to facilitate offline learning. A random start-state and action are selected by Y_f based on the received likelihoods from X_r. The distribution of potential actions for vehicle X is sampled and both the movement of X and Y are simulated by agent Y_f for the entire ΔT using the transition model T defined above.

The end speed for agent Y_f is estimated by assuming that the likelihood of successfully decreasing speed is guaranteed. However, either speeding up or retaining any speed above that of the surrounding traffic, thus reducing the distance between Y and X, may be impeded. For each decision-cycle duration, ΔT, constant acceleration is assumed:

$$\alpha_A = R'_Y - R_Y \tag{4.5}$$

The total decision cycle time is evaluated at a set frequency, $1hz$. Points of interruption are identified by taking the density of vehicles along a section of highway as a uniform distribution and sampling randomly either clear or interrupted for each of the sampled times during δT. At each point of interruption, a non-connected vehicle impedes the motion of the CAV, and the CAV's speed will reset to the average speed of traffic. The final speed is estimated by resuming a constant acceleration α_A for the remainder of the decision cycle.

Values for fuel expenditure and emissions are extrapolated from a table of values to provide a suitable estimate for rewards in each case of simulated experience.

Negotiations

After a preset number of synthetic decision-cycles are completed, the values for each agent's action probabilities, C_x are calculated from the updated set of utilities by transforming the relative values for each action into a probability of that action via a softmax calculation as in Eqn.4.6:

$$
\begin{aligned}
P_a^i &= \frac{\exp \sum Q(s, a_{C_x})}{\sum \exp \sum Q(s, a_{C_x})} \\
Q &= \text{State-Action Value} \\
a &= \text{Action} \\
s &= \text{State} \\
i &= \text{Agent Number}
\end{aligned}
\tag{4.6}
$$

These updated action probabilities inform the the packaged TL grid likelihood values according to Eqn.2.4. The new values are re-broadcast to each neighboring vehicle. This exchange is referred to as a single "negotiation". A series of these negotiations are performed. Over time, the action likelihood values, broadcast TL values, and the utilities converge. The mechanism by which this convergence takes place is not investigated in this work. However, consistent convergence is demonstrated by experimentation. Once either the values have converged or a time threshold is exceeded, the agents will pause negotiations to both record a physical decision cycle and choose the next action.

This complete cycle results in the solution for agent Y_f which learns to optimally choose speeds for fuel economy and emissions accounting for the potential actions of the rear-facing agent X_r. Each vehicle who is trailing another has an active following game which is identical in form. In this example, Z also has a following game in which Z_f will engage with Y.

4.1.2 Leading Game, Y_r

The following game played by Y_f enables vehicle Y to react to actions made by X. In order to complete the network of vehicles, agent Y_r plays a second, "leading game" with Z_f as

in figure 4.1. In this game, agent Y_r splits the distance between Y and Z into identical grids, $\beta_{(Y,Z)}$. The state space for a leading game is comprised of the speed state space of the leading agent Y_r, S_{Y_v}, and the grid position of Z_f. This means that the grid position is defined by the number of grids in between Z and Y. The grids, therefore, are identical in the following-game played by Z_f. The difference between game FG_ZY and LG_YZ is the speed-component of the state-definition. The speed of the game board $\beta_{(Z,Y)}$ is defined by the speed of Y_r.

The action set for Y_r is identical to the potential speed regimes for vehicle Y. The simulated experience steps are calculated by exchanging TL information between Z_f and Y_r in the same fashion that X_r exchanges information with Y_f. The likelihood of Y_r reaching a speed given any action is identical to the likelihood of Y_f reaching that speed regime. Each agent with a trailing CAV, has a leading game established in an identical fashion i.e. X_r has a leading game which it plays with Y_f.

4.1.3 Choosing a Speed for Vehicle Y

To combine the results from both agents on a single CAV, Y, the Q-values for Y_f and Y_r are simply summed together, and an action is selected according to the epsilon-greedy policy applied to the cumulative Q-values. Alternate approaches to combining these rewards were considered but not explored in this work.

If the following vehicle arrives in the grid closest to the leading vehicle, these two vehicles are considered to be forming a platoon. However, if the following vehicle arrives in the farthest of the 10 grids or never reaches a grid closer than grid 10, the vehicles are said to be "disbanding". In these cases either no platoon existed and the RL agent has deemed that forming a platoon is not justified, or it indicates that the RL agents decided to dissolve an existing platoon.

In the presented framework, agents only need to broadcast the TL probabilities to neighboring agents. This strategy should be scaleable to N vehicles with no or relatively little additional computation time. The data exchange occurs between each agent in parallel. The only source of potential time increase comes from any impact on convergence time for the negotiations. However, this can be avoided by adding a maximum time-out value which

cuts off the negotiations at a certain point. The impacts on early termination on performance were explored.

4.2 Experimental Design

Tests were conducted to analyze differences in behavior between nominal driving, and the presented method for vehicle platooning. This work leverages SUMO, an open-source traffic simulation platform, and Simulink, a modeling and simulation platform from MathWorks. Simulink drives the SUMO environment via a Python API to provide an experimental testbed to investigate platooning behaviors. In a standard SUMO simulation, an XML file is written which contains instructions regarding the rate at which vehicles are injected into a scene, their intended trajectory, and other vehicle-specific settings. The SUMO testbed traffic density and speed data was set based on the NGSim dataset for US highway 101. Speeds and vehicles per lane per hour were adapted to match those recorded in the 45 minutes of data collected for US highway 101. This served as a starting point for variations in density. To create a variety of test cases, a program was written to rewrite this file to alter the traffic density programmatically by manipulating the arrival rate and initial vehicle dispersion. The roadway in SUMO was divided into three sections, entry, middle, and exit, all in a straight line. In each test, an initial set of legacy vehicles was dispersed along all three lengths, and the AI-controlled vehicles (if present) were placed in the entry section. During repeated tests, the initial AI locations remained constant but the legacy vehicles were re-distributed.

Three sets of tests were carried out:

1. Internal model creation
2. 3 AI-controlled vehicles
3. 10 AI-controlled vehicles

First, a set of three AI-controlled vehicles were initialized. A series of 40 simulations were carried out wherein a random set of actions were selected at each time step. This series was divided into 4 sets of 10 simulations wherein a new traffic flow rate, modulated by increasing

arrival probability was set for each of the 4 sets. The rates evaluated were 0.1, 0.22, 0.34, 0.46, 0.58, and 0.7 where 0.46 was informed from the US highway 101 dataset. The goal of this was to extract a model for the speed of each AI-vehicle and the resulting distance between AI-driven vehicles following a selected set of speed regimes.

The action-distance model for the environment generated from the first set of simulations was used in a an AI-controlled vehicle study wherein the same 3 vehicles were selected for AI control. In this case, three strategies for vehicle platooning were evaluated:

1. Set vehicle speeds
2. Negotiations with discrete environment model
3. Negotiations with linear action-distance model

The first method, "Set vehicle speeds" seeks to capture the strategy of declaring a platooning zone or region of the highway in which vehicles should form platoons. In this case, the leading vehicle's speed was set at $65mph$ and the trailing vehicles were set to $75mph$ until they drew close enough to be considered platooned. Then, the SUMO driver model was allowed to take over and determine the speeds to avoid collision. The two Negotiating methods leverage the algorithm described above with the difference lying in the underlying model of the environment which provides the synthetic offline learning. In the first method the agents leverage the discretized intermediary layer, probabilistic model for interruption, and a simple constant acceleration model to determine end state. And, the action-distance model approach, the linear model for actions versus ending speed and distance state is used to predict the result of a set of simulated actions from the two negotiating vehicles.

Finally, a set of ten larger tests were performed using the demand estimated from the highway 101 data and a larger set of 10 ai-controlled negotiating vehicles. In these tests, the discretized underlying action model was used for the negotiating agents, and a set of baseline set of tests were carried out against legacy vehicles rather than the set-speed platooning method. Simply allowing the built-in SUMO vehicle following model to drive each vehicle with a set "desired speed" limit of $65mph$ and the default settings for driver aggressiveness, overtake, and any inherent speed variations.

4.3 Results and Discussion

The experiments to determine relationship between the traffic volume, selected speed actions, and resulting velcity and distance traveled revealed a reasonably strong linear relationship between the requested speed regime and the resulting difference in velocities and distances between agents in the simulation. The relationship appears nearly uniform across all of the simulated traffic flow rates at speeds lower than the surrounding traffic, but could be either logarithmic or simply treated as a piecewise function at higher speeds when the traffic flow rate increases.

The results suggest that, beyond an arrival rate of 0.34, the traffic density present on the roadway prevents any of the AI vehicles from proceeding at the requested speed (figure (4.3). Instead, the vehicles likely proceed at the mean traffic speed for the duration of the decision cycle.

A linear model was fit to the data in order to provide a tool from which distances traveled during each time step under new traffic conditions could be estimated.

This model, figure 4.4, was fit to the mean distance traveled for every unique combination of arrival rate and requested speed. The resulting model had an R^2 of 0.9895. This aligns with the flat relationship at higher arrival rates, and could potentially be improved by separating the arrival rates, relating directly to locally measured traffic density, or by leveraging something closer to a piecewise linear model for lower arrival rates. Perhaps exploring the relationship in the context of the fundamental relationships of traffic flow could provide an even more elegant solution for estimating the movement of agents along a given link.

In the second set of tests, the three methods of controlling the AI vehicles were compared. The reward function used in this case included a value for long-term platooning benefits as a function of the number of vehicles in the platoon and the remaining time for those vehicles on the highway. However, the immediate fuel consumption through the traffic was held at a higher value.

In this case, the total fuel consumed by the AI driven vehicles was less while the traffic arrival rate was below 0.46, figure 4.5. The total fuel consumed by the vehicles throughout

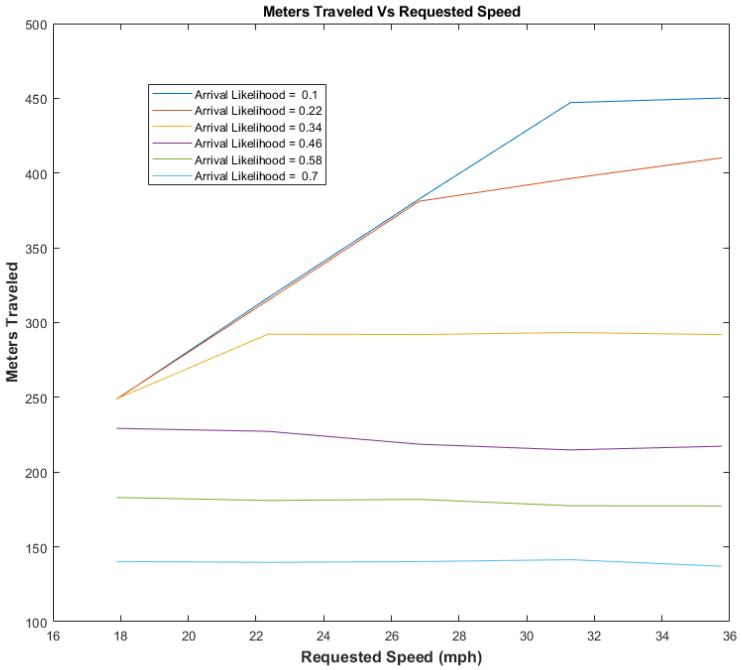


Figure 4.3: Distance vs requested speed for different arrival likelihoods

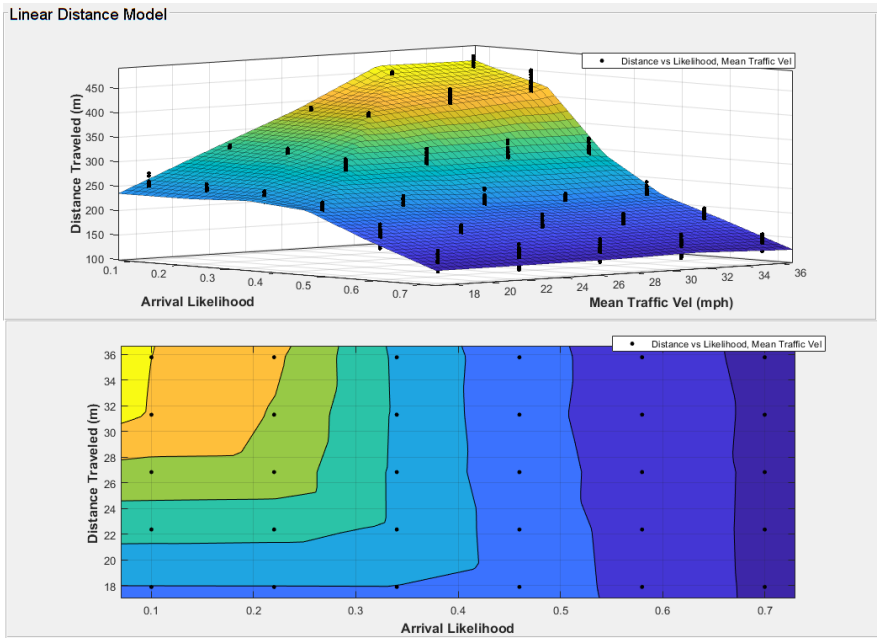


Figure 4.4: Linear distance model

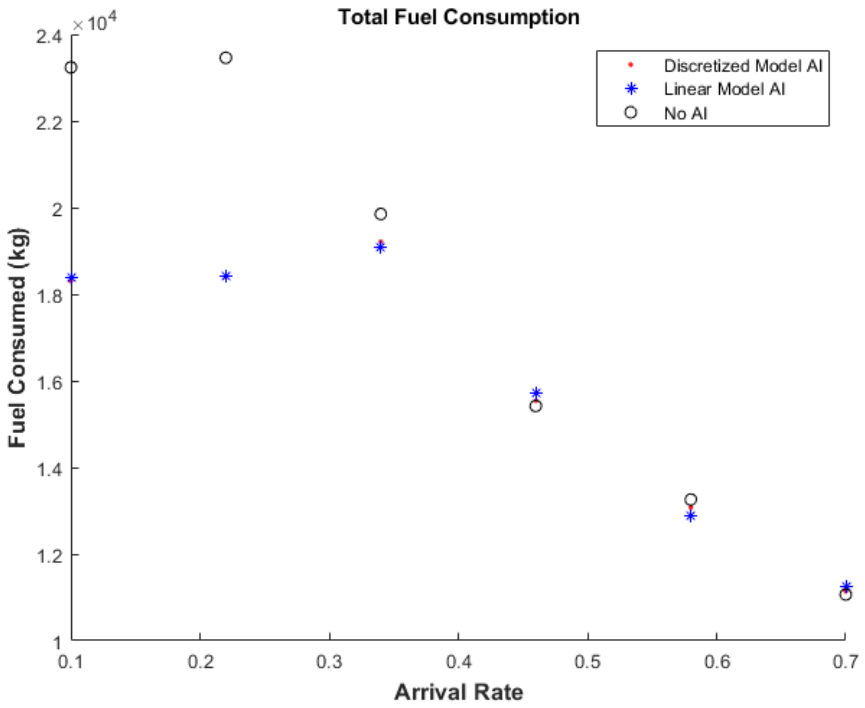


Figure 4.5: Total fuel consumption for the three selected probe vehicles

the simulations for 0.1 and 0.22 vehicle arrival rates was approximately $5000kg$. At 0.34, the difference drops to $640kg$ of fuel, and after this point, the fuel consumption differences between each of the methods was negligible, $< 100kg$. This is likely due to so many other vehicles interrupting or blocking the path of travel for the AI-controlled probe vehicles. Both Fixed-speed and AI-dictated-speed are simply unable to be followed as the surrounding traffic forces the probe vehicles to move at the mean speed. The difference between the two AI controllers was also small, $12.45kg$ on average across each of the simulations. This could indicate that the speed regimes each strategy landed on were identical or close to identical.

One of the ways which this algorithm could be cheating or producing lower fuel consumption is by simply forfeiting speed. If the AI controller was dictating the lowest speed regime for each vehicle, this would lower the fuel consumed. However, the reward function does value maintaining the speed at the desired $65mph$, about $30m/s$. While energy savings are important, it is worth recognizing that drivers could find the lower speeds unpalatable. In this case, the maximum difference in mean vehicle velocities was observed to be $0.5m/s$ in figure 4.6. So, the reinforcement learning negotiations technique is able to reduce the fuel consumption while maintaining a reasonable rate of travel for the probe vehicles.

In the third experiment, 10 probe vehicles were selected from the traffic flow. The goal of this experiment was to see if the benefits were limited to small numbers of vehicles. And, if the fuel savings scale, what is the nature of the relationship?

The rate of fuel consumption for the AI vehicles throughout the simulations was consistently lower, resulting in a lower mean fuel spend as the simulations progressed (figure 4.7). The fuel consumed by the 10 AI controlled negotiating vehicles was $21735kg$ on average less than that of the 10 legacy controlled vehicles. This the mean culmination of fuel savings across all ten heavy duty vehicles for the duration of the simulation. This test was run with the US highway 101 calibrated legacy vehicle arrival rate and repeated 10 times before averaging.

An example single simulation set of speeds is shown in figure 4.8. While most negotiating and legacy vehicles settle on traveling in the range between $55mph$ and $70mph$, there are a few noteworthy differences. Firstly, the lead legacy vehicle increases in speed under legacy

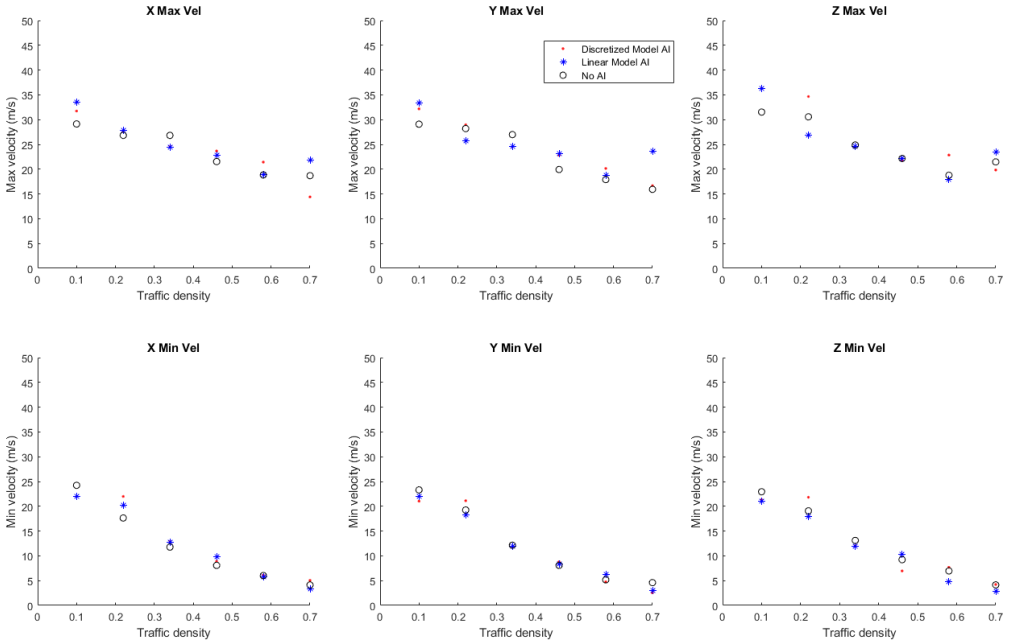


Figure 4.6: Max and min velocities experienced by each of the probe vehicles, X, Y, and Z.

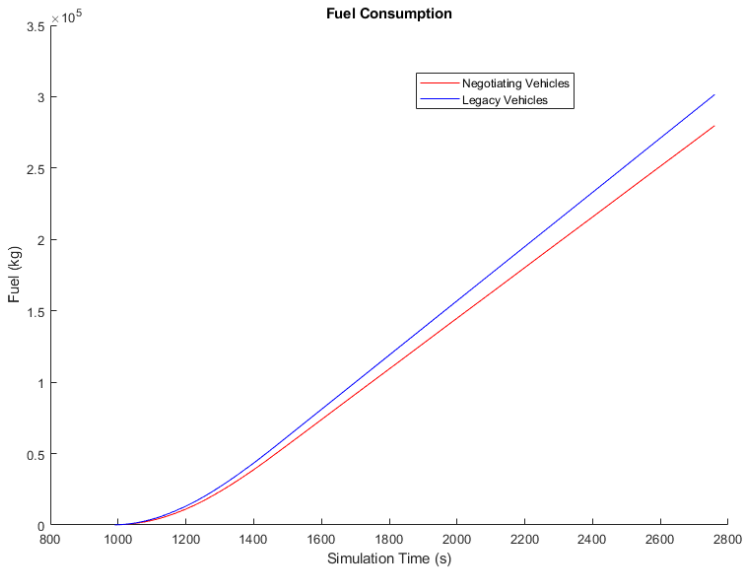
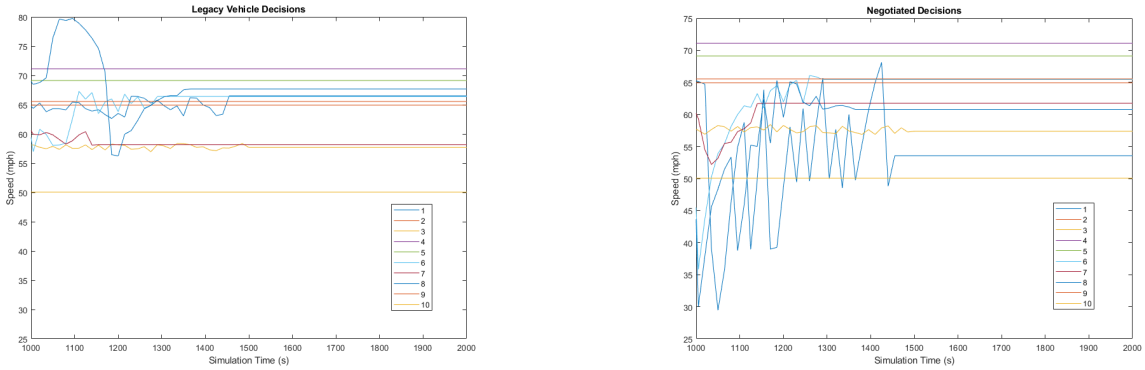


Figure 4.7: Mean fuel savings for 10 vehicles



(a) Probe vehicle speeds under legacy control

(b) Prove vehicle speeds under AI control

Figure 4.8: Cumulative throughput benefit Negotiating - Actuated

control according to the driver model present in the SUMO simulation. Second, the lead AI vehicle slows down dramatically and engages in some erratic behavior until settling at a lower speed. The sawtooth decision making likely cost the negotiations-driven vehicle in terms of both passenger experience and fuel savings. Although smoothing was applied to make the transition between selected speed regimes less abrupt, the constant state of indecision could be the subject of more study or work to improve.

Chapter 5

Extension to Real World & Future Work

5.1 Negotiations

Exploring more thoroughly the convergence mechanism, carefully evaluating any optimality guarantees, and experimenting with alternate underlying machine learning techniques would help to improve upon the the negotiations strategy proposed in this work.

While the exchange of propensity closely resembles fictitious play, the underlying model for the environment could impact the convergence. Other works have explored and proven the convergence of fictitious play with a linear objective function representation. However, the type of nonlinear representations, e.g Q-values, leveraged in this work and future works should be investigated more thoroughly. Verifying that the convergence criteria for the fictitious play is unchanged when paired with the selected underlying ML algorithm is necessary to proceed with future efforts.

Additionally, the algorithm only provided empirically observed benefits in the grid world test, intersection application, and platooning applications. In the other example games, chicken and climbing, the performance matched typical Q-learning and Dyna-Q implementations. In the platooning application specifically, the negotiations strategy outperformed Dyna-Q in a more complex environment. However, the reason for this benefit is not well understood. The underlying model, which the negotiations strategy uses for

pseudo experience planning steps, could be providing a benefit which would diminish over time. Or, the exchange in propensities could be driving the performance improvements towards the end of the simulation when it appears that a steady state has been reached. Properly characterizing the circumstances under which the negotiations strategy produces improved performance and the underlying mechanism is necessary for any further selection of application areas, refinement of the algorithm, etc..

Finally, in the time since this work began, many novel algorithms for reinforcement learning which leverage data-driven methods i.e. deep reinforcement learning (DRL) have emerged. These data-driven models for long-term reward maximization may provide better results than the simplistic probabilistic or discrete models of the environment presented in this work. Instead of performing pseudo-learning steps with the broadcast propensity values, including some small amount of local data from other agents in an environment could result in improved performance. For example, action propensities from nearby agents could be directly included directly as inputs to the deep networks which underpin the DRL algorithms. Negotiations between agents could also be carried out, but this could exacerbate the issues of convergence and optimality as the DRL agents are more opaque and less explainable than the methods presented in this work.

5.2 ITS Applications

Part of the forward-looking intent of this work was to inspire thoughts of collaborative interactions between traffic signal control and platooning vehicles. Using the lake or ocean of data present in the environment to make better decisions is the goal of many ITS applications. Platooning vehicles could transmit data regarding local traffic conditions to signals or vehicles in the same network. This data exchange could replace the link-wide estimate for traffic density and improve the interruption estimates in the case of platooning. Or, for intersection control, it could remove or help to tune the estimate for vehicle arrival rates at each intersection in a network. Similarly, propensities for the AI-driven vehicles and intersections could be exchanged to provide additional information regarding the disposition of ITS elements in a transportation network.

Regarding the exchange of information, it will be important to consider any networking limitations which this work chose to ignore. In both of the applications explored, the broadcast of the propensity values, although small, took place repeatedly between agents' physical action. In actuality, these exchanges may be limited as other, more safety critical, information could consume the available bandwidth. On the other hand, it may be possible to exchange much more information than what was explored in this work. In either of these cases, a more in-depth exploration of the networking aspects of ITS applications and their impact on the performance of negotiations (or any other strategy reliant on data exchange) is necessary.

5.2.1 Traffic Signals

The intent of this work was the presentation of an algorithm which will be adapted for deployment to a real-world traffic intersection in Chattanooga, TN. In order to make this leap, there are a number of issues which must be addressed. Inclusion of pedestrians and cyclists is beyond the scope of this work, and the target deployment arterial, located at an interstate junction, is predominately vehicle traffic. Instead, the key challenges considered in the move to real-world are:

1. Impact of physical distance between each intersection
2. Instrumentation limitations at each signal
3. Effect of more complex signal configurations
4. Computational and broadcast time constraints
5. Driver response to the presented method

First, the impact of physical distance. The assumptions made in the presented work set the stage for reframing the individual 'intermediary zones' as a large Markov Chain. Reframing this completely as a Markov Chain mitigates any negative impact from the length of the roadway between each agent-controlled signal. As explored in the discussion, the arrival rates leveraged by the intersections to estimate their state and action probabilities could

be supplied by measuring the departure from upstream intersections. Two important elements which will be investigated in future works are both formally defining the transition probabilities for this Markov Chain and accounting for any vehicles lost or added through unobserved inlets and outlets i.e. parking lots, small side-streets, etc.. This relies heavily on the instrumentation at each agent-controlled signal.

This raises the importance of challenge two. The target deployment intersection is monitored with a camera system which is capable of counting the number of vehicles both waiting at a small segment of roadway and passing through an intersection. In other applications where such instrumentation is not available, less-accurate estimates should be possible simply by employing strategies such as observing the length of occupied roadway reported by induction loop detectors. For this work, connected vehicles acting as probes were not explored as the market penetration of such vehicles and, more critically, a uniform means of communicating with them, is not currently high enough to be widely applicable.

An additional critical point of difference is the complexity of each signal. The target deployment arterial intersections contain turn lanes in addition to extremely varied lane and phase arrangements, while the presented work explores the behavior of the RL algorithm on simple, 2-phase signals. Extending this algorithm to a more traditional intersection (figure 3.3) under 8-phase dual ring-barrier signal control requires 8 zones, thus 256 states.

The figure illustrating a more complex arrangement (figure 3.3), is included to indicate the ability and intention to extend the method presented in this paper, and it demonstrates the state arrangement for a physical implementation. The simple configuration provides necessary initial data to examine the performance and computation characteristics of the algorithm and negotiations scheme. However, it should be more easily scalable than some RL counterparts. The algorithm presented exhibited a linear increase in computational time, enabling multiple simulated experience steps and negotiations to take place in this work. To move this algorithm to a real-world arterial, the underlying code will be optimized leveraging frame-based calculations to reduce computation times for each intersection. The integrated code-generation capabilities of MATLAB will be leveraged to automatically create a more computationally efficient C++ code from the prototype which was analyzed in this work ([45]). These two steps should greatly reduce the time to compute, package, and broadcast

data from each individual intersection. Additionally, the pseudo-experience implementation presented which leverages the communicated probabilities to simulate previously unvisited states enables each disparate intersection to fully explore a probabilistic RL environment immediately. This strategy should allow for migration to the more complex environment without incurring performance losses resulting from sparsely explored state-spaces both initially and after major shifts in arrivals.

The impact of human behavior will be investigated and considered during the transition to real-world intersection control. There is concern that in unlikely events, an RL-driven traffic signal controller may dramatically reduce or eliminate service for a low-volume lane or roadway. Mitigation strategies include those similar to that studied by Jin and Ma in 2017 wherein they restrict the agent by only allowing for re-activating of a phase after all phases in a cycle have been activated at least once ([37]). This greatly increases the complexity as it adds dependencies on previously executed actions, requiring a reframing of the RL environment. However, it guarantees that no phase is left. Drivers accustomed to traditional controllers may also seek to learn the phase sequence. However, in the case of an RL controlled agent, the pattern may be too complex to memorize or may not exist given the continuous adaptation of the agent to the changing traffic scenario. This issue will be analyzed as an outcome with no current mitigation strategies planned other than retaining the yellow and 'all-red' phase-components to ensure driver safety.

5.2.2 Platooning

The second adaptation of the negotiations algorithm presented was supervisory control for vehicle platooning. While signal control and platooning are markedly different operations, the concept of data-exchange to improve performance provided a common thread. The selected testbed consisted of a single-lane of platooning vehicles with the potential for interruptions by legacy vehicles. The source of the interruption, from the left or right, was not specified. However, the platooning vehicles were not allowed to change lanes. The only algorithm used for comparison was simplistic, dictating the speeds of each platooning vehicle until they were close enough in proximity. The selected metrics were the SUMO fuel

consumption estimation. Finally, information regarding traffic along the entire test highway was provided to the platooning agents. In this case, future work could focus on:

1. Higher-level control strategy to consider lane-changing or overtaking
2. Exploring alternate ML techniques for comparison
3. Improving on the model of the AI controlled vehicles

In the case presented, the platooning vehicles were disallowed to overtake. Instead of driving smoothly in the case of an interruption, the platooning vehicles were forced to slow down to accommodate legacy vehicle traffic. This is largely a byproduct of the selected state-action infrastructure. Instead of dictating speed regimes, the supervisory agent could simply select between actions of "platoon", "disband", and "do nothing" or another set of similarly high-level control strategies. This further distances the supervisory action selected from any safety critical decision. While this work deferred safetly-critical platooning and avoidance maneuvers to the driver model in SUMO, it would be reasonable to suggest that a dictated speed-regime dramatically different than the mean traffic speed along the highway would pose a hazard to the CAV and surrounding legacy vehicles.

Deferring all lower-level vehicle controls and focusing on the outcome, whether the vehicles form a platoon or not, could also enable the negotiations to consider overtaking and lane changes. As vehicle autonomy improves, there could be benefit to allowing for lateral motion. In this case, the underlying model in the presented algorithm would need to consider the likelihood of interruption given the ability to overtake legacy traffic. For example, considering a four-lane highway, the red legacy vehicles might occupy any of the red highlighted squares relative to the green AI vehicle after a time step, and the green vehicle could encounter conflict at the two gold shaded squares in figure 5.1.

This could be treated as a configuration-space problem where the likelihood of some blocking arrangement of legacy vehicles is calculated given a traffic density, or another model could be used to estimate the forward progress of the AI.

It is worth a reminder that no alternatives, machine learning or otherwise, were explored outside of legacy SUMO control and the two negotiations strategies, discrete-model and

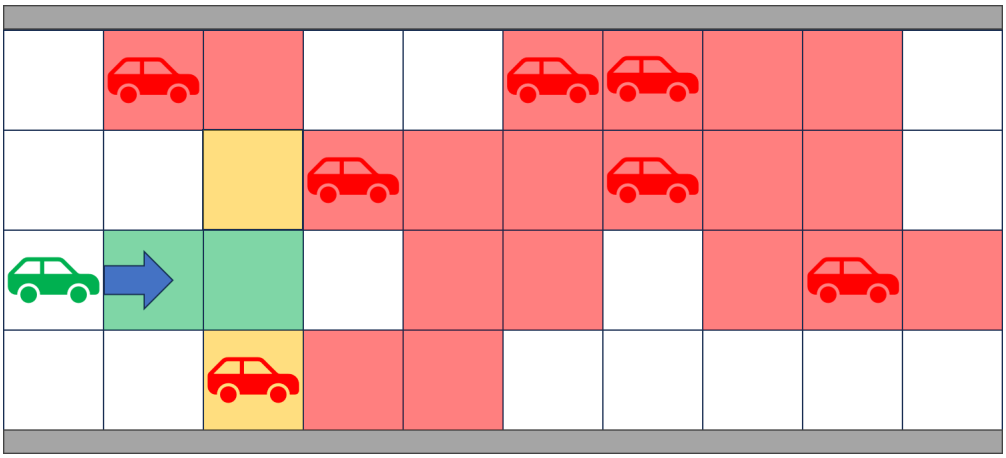


Figure 5.1: Discretized multi-lane traffic navigation example

linear model. What this demonstrates is likely some sort of supervisory control strategy or overall guidance is worth exploring when discussing the application of a platooning region. Distributing a simple Q-learning approach might suffice for scenario investigated in this work. Or, a set of fixed monitors or stations at the entrance to a specified region could recommended vehicles to create or dissolve a platoon based on the current traffic conditions and some supervisory logic or control. Perhaps even a simplistic rules-based approach would suffice.

This work did consider some notion of a long-term benefit term in the reward to encourage platooning based on estimated future fuel savings. However, this number was simply pulled from literature indicating a potential fuel savings given the reduction in drag while in a platoon. The actual fuel consumption recorded was the default estimation provided by SUMO. A more accurate model for the behavior of the AI controlled vehicle, including fuel consumption, could be created within the Simulink and SUMO combined testbed used in this work. Simulink is a modeling and simulation platform used across the automotive industry for the design, testing, and deployment of vehicle control systems [Sim]. Elaborating the vehicle model could provide additional insights into the impact of supervisory control on the fuel consumption of the vehicle. Specifically, the interaction between the supervisory controller and a selected safety-critical controller which must manage the platooning vehicles once the distance is small. Different behavior of AI-controlled vehicle in response to interruptions by legacy vehicles may render the supervisory controller less effective by smoothing the overall trajectory while platoons are forming. However, the impact is difficult to estimate without further study.

Chapter 6

Conclusion

This work introduces a mechanism for MARL agents to arrive at a decision based on the action propensities of their neighbors through an iterative negotiations process. This algorithm includes a combination of several elements:

1. Calculation and broadcast of state probabilities
2. Directly including both action and state probabilities from neighboring agents in the transition probability formulation
3. Providing an underlying 'off-line' learning algorithm which simulates both visited and unvisited states

which, taken together, represent a unique approach to solving a complex coordinated control problem. Then, two applications were explored by casting ITS environments as MARL and applying the negotiations strategy to enable inter-agent coordination.

The traffic signal control implementation builds off of many prior research thrusts. Existing research has explored reinforcement learning as applied to signal timing, grouping movements into phases as actions, and broadcast of action-probabilities. Tests were performed in small scale to demonstrate the computational efficiency of distributing the algorithm across the entire network of intersections. These experiments indicated that scaling the algorithm results in a linear increase in computation time. Additionally, the Negotiations strategy was shown to produce a converged set of action and state propensities in less than

$30sec$ even if the environment is perturbed or the learned utilities are reset. Results from performance tests in a SUMO testbed indicated that the negotiations algorithm increased vehicle throughput, lowered total emissions, and decreased average travel time for a network of signalized intersections. Vehicle throughput was improved by an average of $671veh/hr$ over actuated traffic under static arrival conditions. Travel time through the intersection network, however, was also improved over traditional signal control strategies in static flow, with an average of $9.8sec$ and a maximum of $14.3sec$ saved. In a scenario where traffic arrivals shifted $2hr$ after the simulation start, a maximum emissions reduction of $809kg/hr$ was observed. Despite travel time suffering during the arrival rate shift, the other metrics of success were improved under both test conditions, static arrival rates and shifting arrival rates.

In the case of platooning, the presented framework departs from the bulk of research, which tends to focus on the safety-critical and fine control of platooned vehicle motion; instead, this work chooses to approach platooning from a supervisory angle. In this case, vehicles were allowed to negotiate in order to determine if forming a platoon would be advantageous based on the broadcast action propensities and reward function including long-term fuel consumption benefits in addition to maintaining a desired speed. Through a series of small 3-vehicle and 10-vehicle tests, the maximum observed fuel savings was 20.8% in a low-traffic scenario and closer to 7.2% under realistic traffic scenarios. In each of these cases, the differences in mean speed between legacy driven vehicles and the AI guided vehicles was $< 5m/s$. This combination of reduced fuel consumption and negligible decrease in speed indicate that, rather than simply directing a set of vehicles to platoon, an algorithmic approach to determining when and which vehicles form platoons could be beneficial.

A second outcome of this research was further highlighting the complexity and flexibility in the ITS space as the push for vehicle autonomy and traffic network efficiency continues. The underlying reward scheme and, in fact, much of the core machine learning algorithm used as the means to generate the propensities and perform the negotiations could easily be replaced with more advanced deep neural network architectures. Additionally, the two separate elements, platooning and traffic signal control, could be refactored for interoperation. The sets of negotiating vehicles might be able to act as probe vehicles

to provide more precise information about upcoming traffic flow to the connected grid of signal controllers. These types of system-of-system design challenges, common across many emerging technology areas, increase challenge of design by introducing the issue of scale. This work leveraged MATLAB and Simulink to test and evaluate the proposed MARL algorithm and connected to SUMO to provide the backdrop of synthetic traffic against which the performance of the negotiations was tested. This approach can be made modular and could provide a model-based systems engineering approach to understanding the potential impact of other ITS strategies at a system-of-systems level.

[aut] Autonomous stuff development kits. 66

[urb] Highway and road expenditures. 1

[int] Iei tank* aiot developer kit. 66

[pix] Pixkit: Autonomous driving development kit. 66

[nvi] Self-driving cars technology amp; solutions from nvidia automotive. 66

[Sim] Simulation and model-based design - simulink. 91

[7] (2022). Sources of greenhouse gas emissions. 1

[8] (2024). Architecture reference for cooperative and intelligent transportation$_2$024. 2

[9] Abboud, K., Omar, H. A., and Zhuang, W. (2016). Interworking of dsrc and cellular network technologies for v2x communications: A survey. *IEEE transactions on vehicular technology*, 65(12):9457–9470. 15

[10] Acosta, A. F., Espinosa, J. E., and Espinosa, J. (2015). Traci4matlab: enabling the integration of the sumo road traffic simulator and matlab® through a software re-engineering process. In *Modeling Mobility with Open Data*, pages 155–170. Springer. 38

[11] Arel, I., Liu, C., Urbanik, T., and Kohls, A. G. (2010). Reinforcement learning-based multi-agent system for network traffic signal control. *IET Intelligent Transport Systems*, 4(2):128–135. 7

[12] Ayodele, T. O. (2010). Types of machine learning algorithms. In *New advances in machine learning*. IntechOpen. 3

[13] Bakule, L. (2008). Decentralized control: An overview. *Annual reviews in control*, 32(1):87–98. 6

[14] Balachandar, N., Dieter, J., and Ramachandran, G. S. (2019). Collaboration of ai agents via cooperative multi-agent deep reinforcement learning. *arXiv preprint arXiv:1907.00327*. 6

[15] Balaji, P., German, X., and Srinivasan, D. (2010). Urban traffic signal control using reinforcement learning agents. *IET Intelligent Transport Systems*, 4(3):177–188. 50

[16] Barlow, H. B. (1989). Unsupervised learning. *Neural computation*, 1(3):295–311. 4

[17] Bazzan, A. L. (2009). Opportunities for multiagent systems and multiagent reinforcement learning in traffic control. *Autonomous Agents and Multi-Agent Systems*, 18(3):342. 7

[18] Bearth, D. P. (2019). Us logistics costs rise 11.4% in 2018. 1

[19] Bergenhem, C., Shladover, S., Coelingh, E., Englund, C., and Tsugawa, S. (2012). Overview of platooning systems. In *Proceedings of the 19th ITS World Congress, Oct 22-26, Vienna, Austria (2012)*. 8

[20] Berger, U. (2007). Brown's original fictitious play. *Journal of Economic Theory*, 135(1):572–578. 6, 14, 18

[21] Bishop, C. M. (2006). *Pattern recognition and machine learning*. Springer Science+ Business Media. 3

[22] Blogg, M., Semler, C., Hingorani, M., and Troutbeck, R. (2010). Travel time and origin-destination data collection using bluetooth mac address readers. In *Australasian transport research forum*, volume 36. 1

[23] Boardman, J. and Sauser, B. (2006). System of systems-the meaning of of. In *2006 IEEE/SMC international conference on system of systems engineering*, pages 6–pp. IEEE. 8

[24] Buşoniu, L., Babuška, R., and De Schutter, B. (2010). Multi-agent reinforcement learning: An overview. *Innovations in multi-agent systems and applications-1*, pages 183–221. 11

[25] Claus, C. and Boutilier, C. (1998). The dynamics of reinforcement learning in cooperative multiagent systems. *AAAI/IAAI*, 1998(746-752):2. 21

[26] Cook, F. S. (2004). Vehicle traffic monitoring using cellular telephone location and velocity data. US Patent 6,810,321. 1

[27] Faragó, P., Cîrlugea, M., and Hintea, S. (2021). Coordination game analysis using brown's fictitious play. In *2021 44th International Conference on Telecommunications and Signal Processing (TSP)*, pages 175–178. IEEE. 3

[28] Foerster, J., Chen, R. Y., Al-Shedivat, M., Whiteson, S., Abbeel, P., and Mordatch, I. (2018). Learning with opponent-learning awareness. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, pages 122–130. International Foundation for Autonomous Agents and Multiagent Systems. 6

[29] Genders, W. and Razavi, S. (2016). Using a deep reinforcement learning agent for traffic signal control. *arXiv preprint arXiv:1611.01142*. 7

[30] Genders, W. and Razavi, S. (2018). Evaluating reinforcement learning state representations for adaptive traffic signal control. *Procedia computer science*, 130:26–33. 7

[31] Geroliminis, N. and Daganzo, C. F. (2008). Existence of urban-scale macroscopic fundamental diagrams: Some experimental findings. *Transportation Research Part B: Methodological*, 42(9):759–770. 15

[32] Gupta, J. K., Egorov, M., and Kochenderfer, M. (2017). Cooperative multi-agent control using deep reinforcement learning. In *International Conference on Autonomous Agents and Multiagent Systems*, pages 66–83. Springer. 6

[33] Howard, R. A. (1960). Dynamic programming and markov processes. 5

[34] Hunt, P., Robertson, D., Bretherton, R., and Winton, R. (1981). Scoot-a traffic responsive method of coordinating signals. Technical report. 7

[35] Hwang, K.-S., Jiang, W.-C., and Chen, Y.-J. (2014). Model learning and knowledge sharing for a multiagent system with dyna-q learning. *IEEE transactions on cybernetics*, 45(5):978–990. 5

[36] Jie, L., Van Zuylen, H., Chunhua, L., and Shoufeng, L. (2011). Monitoring travel times in an urban network using video, gps and bluetooth. *Procedia-Social and Behavioral Sciences*, 20:630–637. 1

[37] Jin, J. and Ma, X. (2017). A group-based traffic signal control with adaptive learning ability. *Engineering applications of artificial intelligence*, 65:282–293. 88

[38] Karhunen, J., Raiko, T., and Cho, K. (2015). Unsupervised deep learning: A short review. *Advances in independent component analysis and learning machines*, pages 125–142. 4

[39] Karkhanis, P., van den Brand, M. G., and Rajkarnikar, S. (2018). Defining the c-its reference architecture. In *2018 IEEE International Conference on Software Architecture Companion (ICSA-C)*, pages 148–151. IEEE. 2

[40] Kavathekar, P. and Chen, Y. (2011). Vehicle platooning: A brief survey and categorization. In *ASME 2011 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, pages 829–845. American Society of Mechanical Engineers. 8

[41] Knospe, W., Santen, L., Schadschneider, A., and Schreckenberg, M. (2002). A realistic two-lane traffic model for highway traffic. *Journal of Physics A: Mathematical and General*, 35(15):3369. 15

[42] Kok, J. R. and Vlassis, N. (2006). Collaborative multiagent reinforcement learning by payoff propagation. *Journal of Machine Learning Research*, 7(Sep):1789–1828. 6

[43] Kotsiantis, S. B., Zaharakis, I., and Pintelas, P. (2007). Supervised machine learning: A review of classification techniques. *Emerging artificial intelligence applications in computer engineering*, 160:3–24. 4

[44] Krishna, V. and Sjöström, T. (1998). On the convergence of fictitious play. *Mathematics of Operations Research*, 23(2):479–511. 18

[45] Krizan, J., Ertl, L., Bradac, M., Jasansky, M., and Andreev, A. (2014). Automatic code generation from matlab/simulink for critical applications. In *2014 IEEE 27th Canadian Conference on Electrical and Computer Engineering (CCECE)*, pages 1–6. IEEE. 87

[46] Kuvayev, L. and Sutton, R. S. (1996). Model-based reinforcement learning with an approximate, learned model. In *Proceedings of the ninth Yale workshop on adaptive and learning systems*, pages 101–105. Citeseer. 5

[47] Li, L., Lv, Y., and Wang, F.-Y. (2016). Traffic signal timing via deep reinforcement learning. *IEEE/CAA Journal of Automatica Sinica*, 3(3):247–254. 7

[48] Lopez, P. A., Behrisch, M., Bieker-Walz, L., Erdmann, J., Flötteröd, Y.-P., Hilbrich, R., Lücken, L., Rummel, J., Wagner, P., and Wießner, E. (2018). Microscopic traffic simulation using sumo. In *The 21st IEEE International Conference on Intelligent Transportation Systems*. IEEE. 38

[49] Lorena (2022). Transportation costs for businesses: The breakdown. 1

[50] Lu, X.-Y. (2018). Truck cacc implementation, field test and data analysisresults. 8

[51] Maier, M. W. (1998). Architecting principles for systems-of-systems. *Systems Engineering: The Journal of the International Council on Systems Engineering*, 1(4):267–284. 8

[52] Marsland, S. (2014). *Machine learning: an algorithmic perspective*. Chapman and Hall/CRC. 3

[53] McQueen, B. and McQueen, J. (1999). *Intelligent transportation systems architectures*. 8

[54] Medina, J. C. and Benekohal, R. F. (2012). Traffic signal control using reinforcement learning and the max-plus algorithm as a coordinating strategy. In *2012 15th International IEEE Conference on Intelligent Transportation Systems*, pages 596–601. IEEE. 38, 47

[55] Melo, F. S. (2001). Convergence of q-learning: A simple proof. *Institute Of Systems and Robotics, Tech. Rep*, pages 1–4. 5, 19

[56] Michon, J. A. (1985). A critical view of driver behavior models: what do we know, what should we do? *Human behavior and traffic safety*, pages 485–524. 15

[57] Milanés, V., Shladover, S. E., Spring, J., Nowakowski, C., Kawazoe, H., and Nakamura, M. (2013). Cooperative adaptive cruise control in real traffic situations. *IEEE Transactions on Intelligent Transportation Systems*, 15(1):296–305. 8

[58] Miller, J. (2008). Vehicle-to-vehicle-to-infrastructure (v2v2i) intelligent transportation system architecture. In *2008 IEEE intelligent vehicles symposium*, pages 715–720. IEEE. 9

[59] Mitchell, T. M. (1997). *Machine learning*. MacGraw-Hill. 1, 3

[Norm] Norm, I. Iec 15622: 2002 (e) 2002. *Transport Information and Control Systems–Adaptive Cruise Control Systems–Performance Requirements and Test Procedures*. 8

[61] Osborne, M. J. and Rubinstein, A. (1994). *A course in game theory*. MIT press. 6

[62] Panait, L. and Luke, S. (2005). Cooperative multi-agent learning: The state of the art. *Autonomous agents and multi-agent systems*, 11(3):387–434. 2, 5, 7

[63] Płaczek, B. (2018). A hierarchical cellular automaton model of distributed traffic signal control. *arXiv preprint arXiv:1809.10892*. 7

[64] Ploeg, J., Scheepers, B. T., Van Nunen, E., Van de Wouw, N., and Nijmeijer, H. (2011). Design and experimental evaluation of cooperative adaptive cruise control. In *2011 14th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, pages 260–265. IEEE. 8

[65] Qureshi, K. N. and Abdullah, A. H. (2013). A survey on intelligent transportation systems. *Middle-East Journal of Scientific Research*, 15(5):629–642. 1

[66] Rajkumar, S., Prethi, V., and Priyadharshini, B. (2017). Traffic signal control using web-camera and traffic flow estimation. In *Proceedings of the International Conference on Intelligent Computing Systems (ICICS 2017–Dec 15th-16th 2017) organized by Sona College of Technology, Salem, Tamilnadu, India*. 1

[67] Rummery, G. A. and Niranjan, M. (1994). *On-line Q-learning using connectionist systems*, volume 37. University of Cambridge, Department of Engineering Cambridge, England. 5

[68] Saradha, B. J., Vijayshri, G., and Subha, T. (2017). Intelligent traffic signal control system for ambulance using rfid and cloud. In *2017 2nd International Conference on Computing and Communications Technologies (ICCCT)*, pages 90–96. IEEE. 1

[69] Sheard, S. A. and Mostashari, A. (2009). Principles of complex systems for systems engineering. *Systems Engineering*, 12(4):295–311. 8

[70] Siljak, D. D. (2011). *Decentralized control of complex systems*. Courier Corporation. 6

[71] Sims, A. G. and Dobinson, K. W. (1980). The sydney coordinated adaptive traffic (scat) system philosophy and benefits. *IEEE Transactions on vehicular technology*, 29(2):130–137. 7

[72] Sutton, R. S., Barto, A. G., et al. (1998). *Introduction to reinforcement learning*, volume 2. MIT press Cambridge. 3, 5, 12

[73] Sutton, R. S., Szepesvári, C., Geramifard, A., and Bowling, M. P. (2012). Dyna-style planning with linear function approximation and prioritized sweeping. *arXiv preprint arXiv:1206.3285*. 5

[74] Szepesvári, C. and Littman, M. L. (1999). A unified analysis of value-function-based reinforcement-learning algorithms. *Neural computation*, 11(8):2017–2060. 3

[75] Valmari, A. (2005). The state explosion problem. *Lectures on Petri Nets I: Basic Models: Advances in Petri Nets*, pages 429–528. 3

[76] van der Pol, E. (2016). Deep reinforcement learning for coordination in traffic light control. *Master's thesis, University of Amsterdam*. 7

[77] Wang, W., Hao, J., Wang, Y., and Taylor, M. (2019). Achieving cooperation through deep multiagent reinforcement learning in sequential prisoner's dilemmas. In *Proceedings of the First International Conference on Distributed Artificial Intelligence*, page 11. ACM. 6

[78] Watkins, C. J. and Dayan, P. (1992). Q-learning. *Machine learning*, 8(3-4):279–292. 5

[79] Watkins, C. J. C. H. (1989). Learning from delayed rewards. 12

[80] Watson, M., Anway, R., McKinney, D., Rosser, L. A., and MacCarthy, J. (2019). Appreciative methods applied to the assessment of complex systems. In *INCOSE International symposium*, volume 29, pages 448–477. Wiley Online Library. 8

[81] Webster, F. V. (1958). Traffic signal settings. Technical report. 7

[82] Wongpiromsarn, T., Uthaicharoenpong, T., Wang, Y., Frazzoli, E., and Wang, D. (2012). Distributed traffic signal control for maximum network throughput. In *2012 15th international IEEE conference on intelligent transportation systems*, pages 588–595. IEEE. 7

[83] Xiao, L. and Gao, F. (2010). A comprehensive review of the development of adaptive cruise control systems. *Vehicle system dynamics*, 48(10):1167–1192. 8

[84] XING, S.-Y., LIAN, G.-L., YAN, D.-Y., and CAO, J.-Y. (2018). Traffic signal light optimization control based on fuzzy control and ccd camera technology. *DEStech Transactions on Computer Science and Engineering*, (cmsms). 1

[85] Yau, K.-L. A., Qadir, J., Khoo, H. L., Ling, M. H., and Komisarczuk, P. (2017). A survey on reinforcement learning models and algorithms for traffic signal control. *ACM Computing Surveys (CSUR)*, 50(3):34. 7

[86] Yousef, K. M., Al-Karaki, M. N., and Shatnawi, A. M. (2010). Intelligent traffic light flow control system using wireless sensors networks. *J. Inf. Sci. Eng.*, 26(3):753–768. 7

[87] Zantalis, F., Koulouras, G., Karabetsos, S., and Kandris, D. (2019). A review of machine learning and iot in smart transportation. *Future Internet*, 11(4):94. 1

[88] Zhang, L., Chen, F., Ma, X., Pan, X., et al. (2020). Fuel economy in truck platooning: A literature overview and directions for future research. *Journal of Advanced Transportation*, 2020. 8

[89] Zhao, D., Dai, Y., and Zhang, Z. (2011). Computational intelligence in urban traffic signal control: A survey. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 42(4):485–494. 7

[90] Zhu, F., Aziz, H. A., Qian, X., and Ukkusuri, S. V. (2015). A junction-tree based learning algorithm to optimize network wide traffic control: A coordinated multi-agent framework. *Transportation Research Part C: Emerging Technologies*, 58:487–501. 7

[91] Zhu, L., Yu, F. R., Wang, Y., Ning, B., and Tang, T. (2018). Big data analytics in intelligent transportation systems: A survey. *IEEE Transactions on Intelligent Transportation Systems*, 20(1):383–398. 1

[92] Zou, L., Xia, L., Du, P., Zhang, Z., Bai, T., Liu, W., Nie, J.-Y., and Yin, D. (2020). Pseudo dyna-q: A reinforcement learning framework for interactive recommendation. In *Proceedings of the 13th International Conference on Web Search and Data Mining*, pages 816–824. 3, 5, 15

Vita

Russell Graves is a native of east Tennessee, born and raised in Knoxville. He attended the Farragut public school system until his graduation from High School in Spring of 2009. Russell's interest in engineering built from role models like his father and a number of key influential figures in his secondary education in addition to exposure to STEM hobbies. In the Fall of 2009, Russell entered the University of Tennessee with the goal of achieving a degree in Mechanical Engineering. By Spring 2013 he graduated, and would fatefully meet Dr. Chakraborty, who employed him as a graduate research assistant through his Masters of Mechanical Engineering in Spring of 2017. After completing his coursework and beginning to compile his dissertation, Russell went on to accept a position at MathWorks in Natick, MA. He completed his doctoral studies, receiving a PhD in Mechanical Engineering in the Spring of 2024.