Reinforcement Learning of a Four-Way Traffic Signal System

Joe Mockler
Department of Aerospace Engineering
University of Maryland, College Park
College Park, MD
jmockle1@umd.edu

***Abstract*—This work demonstrates the application of a reinforcement-learning algorithm to a signalized intersection. A topical overview of more conventional analytical techniques used by designers today is first presented. Then, an MDP is constructed for a typical simple intersection with some assumptions made to ensure solve-ability. The convention analysis is applied to this MDP to serve as a baseline to compare reinforcement learning results to. A maximum likelihood RL algorithm is deployed to solve the MDP and determine an optimal policy given intersection characteristics. This learned policy significantly outperforms the conventional analysis by minimizing queue lengths amongst all approaches; the primary mechanism the signal switching more often to avoid vehicle accumulation. While this construction shows promise, typical intersections have many phases and assuming the total vehicles at an intersection is not always possible. Future work should focus on handling these limitations as the state-of-the-art tries to solve a more representative intersection model.**

A note on paper formatting: the sections ”Overview, Background, Methods, Results, Discussion, Future Work” are all addressed in respective order throughout the paper, just not by name (because I wrote the paper prior to seeing the canvas notification about the structure). Please note that ”Overview” is addressed in Introduction, ”Background” in Background and Traffic Flow and Signaling Theory, and finally, ”Methods” in Markov Decision Process and Maximum Likelihood RL. The other sections are addressed by name.

***Index Terms*—Signal timing, Signalized intersections, Traffic engineering, Reinforcement learning, Maximum likelihood**

I. Introduction

Nearly all interconnected traffic routes require a form of traffic control; in moderately congested traffic routes, the traffic light has become ubiquitous and the most effective traffic controller. These signaling systems are incredibly safe and deployed across the U.S; to deploy, designers must determine the red-green-yellow lights in a cycle. At the heart of this design problem is determining the optimal time for a phase – exactly which light should be green/red and for how long?

II. Background

The industry standard for cycle timing is capacity and critical movement analysis [1]. In this technique, an intersection's capacity and expected volume are first estimated using measurement data or forecasted with other methods for each critical movement (ex: a Northbound left, a Southbound right, an eastbound straight). The light cycle timings are then allocated according to road geometry, direction of traffic, time of day, and expected traffic flow.

This analysis obviously falls short when estimates are incorrect and may produce sub-optimal results. Mismatch between estimate and practice are costly: unnecessary delays add in travel/shipping costs, reduce safety, increase emissions, and consume energy. In some major cities, timing mismatch costs millions of dollars each year [4]. To combat this inherent flaw, some advanced techniques have been deployed in the past three decades, including phase recalls and improved simulation/time-of-day cycle modelling. In phase recalling, a vehicle is detected, and the current cycle's phase is interrupted to accommodate detected traffic [3]. These detection techniques include induction sensors, pressure sensors, infrared sensors, or camera/computer vision systems [2].

Extending the advancements of vehicle detection techniques, reinforcement learning, coupled with a suitable traffic model, offers a potential for improved traffic flow without relying on estimates. These RL techniques offer an advantage of being able to learn a policy based on the exact state of the system rather than a preconceived estimate. While RL techniques were first applied to this problem in 2011, the problem is finally getting more traction and seeing a variety of approaches taken [5], [6].

In this work, a reinforcement algorithm is applied to a traffic signal planning problem. First, some background is introduced to familiarize the reader with typical traffic signaling theory and flow. Next, an MDP will be constructed around a 4-way, signalized intersection, drawing heavily from the theory introduced in the previous section. Next, the reinforcement algorithm is introduced and its implementation detailed. Finally, the results of the learned policy are discussed relative to typical signaling theory, where the author will demonstrate a marked improvement with the learned policy. Finally, some potential shortcomings and future research directions will be summarized.

III. Traffic Flow and Signaling Theory

This section details a brief topical overview of convention methods of signal timing at intersections. To do so, some basic traffic flow theories are introduced followed by an analysis of signal timing at intersections.

A. Signalized Traffic Flow: Basics

The goal of the signal is two-fold: 1) reduce the probability of crashes and 2) minimize the wait time or delay in an intersection. These objectives are achieved by minimizing the possible conflict points when assigning the right of way to different traffic streams at different times [7]. As a designer, the ultimate goal is to pick which lanes are permitted in the intersection for a prescribed amount of time. The most prevalent theory makes use of *capacity and critical movement analysis*, where the lanes permitted to move are first selected, then the amount of time is prescribed to each moving lane. Some important definitions are first outlined below [7]:

- **Lane group**: the group under a shared stop or go, e.g. all lanes on a northbound flow, left-turn lanes, or cross-traffic. Note that this may include pedestrian cross-walks in high foot traffic areas.
- **Phase**: part of a cycle allocated to a stream of traffic or a combination of two or more streams of traffic having the right-of-way.
- **Cycle**: the time in seconds required for one complete color sequence of signal indication.
- **Saturation flow,** s_i: the flow rate (typically in veh/time) that the lane group can carry if the signal turns green and queued traffic is emptied. Note the relationship: saturation flow = capacity / green time.
- **Approach flow,** q_i: the flow rate incoming on a lane group (sometimes referred to discharge rate).

Let's visualize this with a typical flow diagram in Fig. 1. As the cycle turns green, there is an initial small period (loss time) as the flow increases to saturation flow. The signal then clears the intersection queue and tapers to the approach flow. After the phase ends, the signal turns yellow, then red, and the next phase begins.

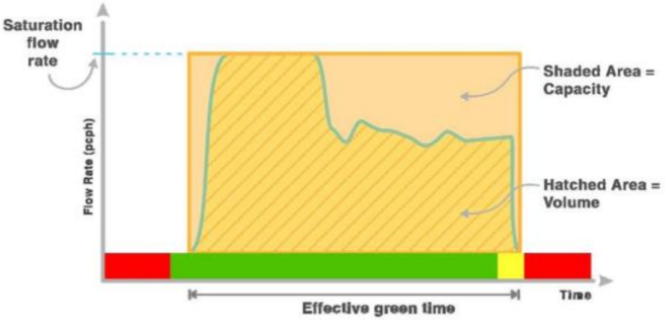

Fig. 1. A typical diagram of flow from a lane group when a phase turns green. The saturation flow is the dense flow that follows the indicator turning green. When the queue clears, the flow tapers to only letting the approach flow through the intersection.

This type of flow occurs for each lane group in an intersection. Fig. 2 helps visualize some of these definitions and how the flows at intersection relate. In this diagram, lane groups are lanes that would move simultaneously on a green, like the N/S left-turn lanes. The depicted phase is allowing the N/S left-turns to move and holds the other lane groups as red.

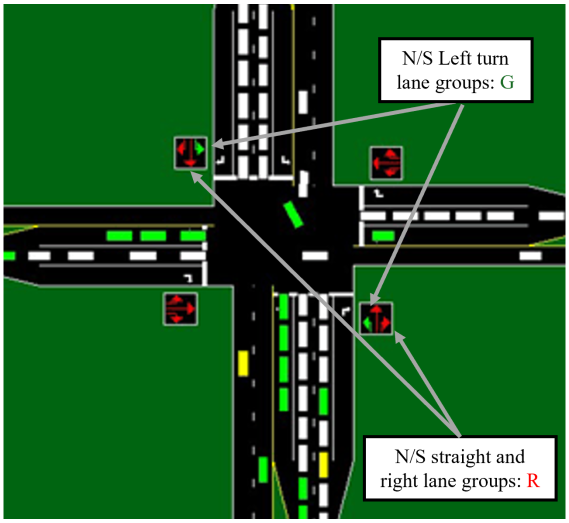
N/S Left turn lane groups: G
N/S straight and right lane groups: R

Fig. 2. A typical phase intersection undergoing a cycle. In this diagram, the N/S left-turn lane groups are permitted during this phase. This graphic is taken from a typical traffic planning software used in civil engineering applications

B. Capacity and Critical Movement Analysis

In the first step of timing design, phases are first identified by the designer, typically done as sees fit. For example, cross-traffic flow and main traffic flow may each receive a phase. In addition, if there are pedestrians or significant left-turn traffic, they will each receive a phase. Next, the cycle length is estimated using the *critical ratio* of an intersection

$$X_i = \sum_i \left(\frac{q_i}{s_i}\right)\left(\frac{C}{C-L}\right) \tag{1}$$

where X_i is the critical ratio for an intersection and L is the total lost time (sum of all-red phases and loss time at start of greens) [7]. When $X_i = 1.00$, the intersection is at critical capacity; over 1.00 and a lane group risks becoming over-saturated, where vehicles cannot be sufficiently cleared during the light, i.e. *cycle failure*. Typically, designers will select $X_i \approx 0.9$ to provide a factor-of-safety. Knowing X_i, q_i, s_i, and L, the total cycle length C may be solved for.

Next, the allocation of green times to each phase are estimated. The total effective green time is first calculated as the cycle length minus loss time, then the green are allocated based on the maximum ratio of approach to saturation flows under a lane group (i.e. for a lane group with multiple approaches the ratio is computed for *each* approach and the maximum is taken) [7].

$$G_{te} = C - \left(\sum_{i=1}^{\phi} l_i + R\right) \tag{2}$$

$$G_{ei} = \frac{\max_j(q_{ij}/s_j)}{\sum_{i=1}^{\phi}\max_j(q_{ij}/s_j)} G_{te} \tag{3}$$

So far, the q_i and s_i are assumed to be known, but some comments on this assumption are required. s_i is usually calculated as a function of road design and regulation (e.g. speed limits) and is usually well-assumed [7]. If the traffic pattern is new, q_i must be estimated from existing traffic flow and forecasts. If a signal is being re-timed, then q_i comes from traffic measurements [3]. Therefore, q_i is usually an uncertain factor in these calculations.

While there are increasing sophisticated models for signal timing and analysis (including coordinated timing, split-phase timing, signal actuation, and scalar adjustments), this method is a typical starting point for further design or early implementation on new intersection [2]. A common practice is create different cycle based on different flows, which vary by time of day, day of week, and by month. In fact, many intersection have dozens of different cycle designs, all carefully tuned to estimated traffic demands [1], [2]. An obvious result of this is an over-dependence on flow estimates; these methods are not usually robust to estimation uncertainties and may fail for unexpected flows. This is explored in later sections.

IV. Markov Decision Process

This section will detail the framing of a four-way signalized intersection as a Markov decision process (MDP). Later analysis will exploit this framework to design a solver that may learn a cycle.

A. Process Modelling

First, the components of an MDP will be defined. Consider Fig. 3, which will compose the basis of this work. Let's first assume that vehicles may only continue straight. Turns would create a time-varying s_i or require more lane groups and complicated transitions, making modeling complicated. Next assume that the vehicles are all an average length, L, and a time is found such that at least a few vehicles can clear the intersection in one time step, thus ensuring the space is discrete. Finally, let's assume that the number of vehicles in a queue can be reasonably counted or estimated [3].

The traffic model MDP is defined as:

- **States**: The states are a tuple representing the queue lengths on each approach and the current signal (green on NS or green on EW): $S := (l_N, l_S, l_E, l_W, \phi)$
- **Action**: The action is the current phase. Either N/S traffic is let through (green on N/S, red on E/W) or E/W traffic is let through: $A := (:NS, :EW)$
- **Transition**: Under a single time-step, t, the transition will let the s_i vehicles through the phase lane groups under the indicated phase while accumulating q_i vehicles under all other phases. If the phase changes, then an all-red phase accumulates variables for two time steps.

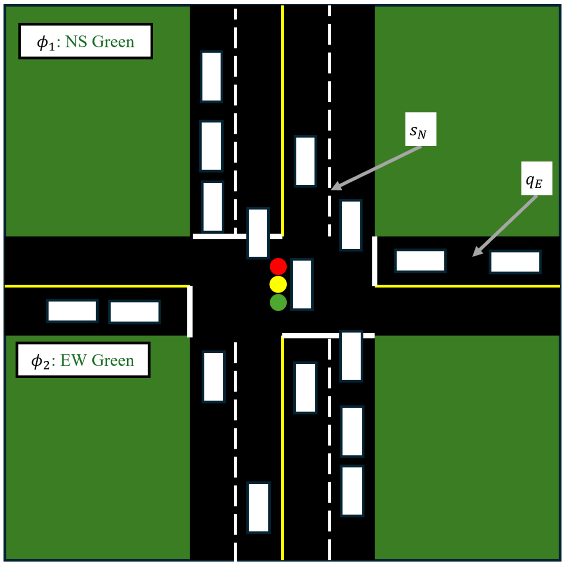

Fig. 3. The traffic geometry that is assumed by the MDP. Depicted is the traffic currently in phase ϕ_1, where the N/S flows are moving while the E/W flows are accumulating. A few traffic characteristics are called out.

$$S' = \begin{cases} T(s'|s, :NS) = S + (-s_N, -s_S, +q_E, +q_W) \\ \qquad\qquad\qquad \text{if S(5) = :NS} \\ T(s'|s, :EW) = S + (+q_N, +q_S, -s_E, -s_W) \\ \qquad\qquad\qquad \text{if S(5) = :EW} \\ T(s'|s, a) = S + 2(q_N, q_S, q_E, q_W) \\ \qquad\qquad\qquad \text{if S(5)} \neq \text{a} \end{cases} \tag{4}$$

Note that $S \in \mathbb{W}$.

- **Reward**: The reward is the negative of the total queue lengths: $R(s, a) = -||S(1:4)||_1$ If any queue length reaches a defined upper limit, then $R(s, a) = -1,000$ to give a strong, negative penalty.
- **Discount,** γ: 0.99

Under this model, it is assumed the saturation flow rate is both known and deterministic (reasoned in sec. III.B) The accumulation rate is stochastic and modelled as a Poisson distribution with mean μ_i. Therefore, the approach flows are sampled from the following distribution:

$$q_i \sim \frac{\mu_i^k}{k!} e^{\mu_i} \tag{5}$$

This MDP also assumes that flow will continue through the yellow phase and can be lumped into the green phase (i.e. there will be no yellow phase). In sec. V. a reinforcement learning algorithm will exploit this construction to learn a cycle.

B. Capacity and Critical Movement: Application

As a baseline, capacity and critical movement analysis is applied to the constructed MDP. This will ultimately serve as a comparison against the learned policy in sec. V.

First, the four approaches are grouped into two phases: ϕ_1 is a north/south green and east/west red; ϕ_2 is an east/west green and north/south red. For this example, the N/S lanes are in higher demand while E/W are secondary roads. The flow-rates are populated in the table I. For design purposes, $s_i = \mu_i$, which is a typically acceptable calculation to avoid continued sampling from the distribution. Applying (1) and setting $X_i = 0.85$ and $L = 2$, the total cycle length is 18 time intervals, commonly rounded to the nearest 5. Now, with a total cycle time $C = 20$, (2) and (3) are used to estimate the green time per phase. Assuming there is no start-up loss (or it is minimal compared to the total green time), $G_{te} = 18$ and the NS phase is allotted 10 time intervals while the EW is given 8.

As example, the MDP constructed in part A is simulated with this policy. For the first 10 steps of the cycle, a = :NS, then the signal switches (accumulating two time-steps of vehicles during the modelled ”all red”), and finally, a = :EW for 8 more time steps. The resulting flow appears stable over 1000 time steps.

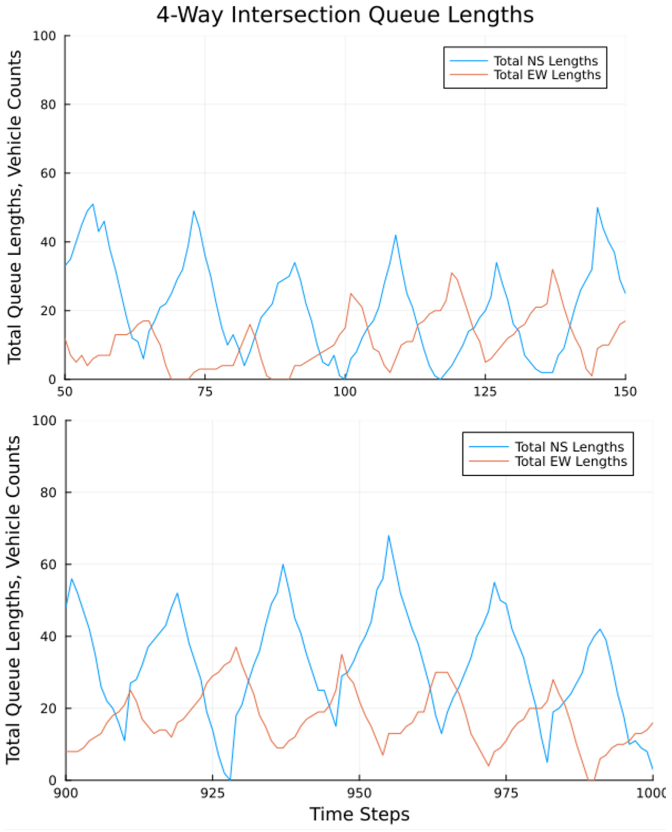

Fig. 4. Following the critical capacity and movement analysis, a signal system is constructed and deployed on the MDP. The queue lengths remain largely stable over 1000 simulated time steps.

As another example, say a failing sensor monitoring an intersection artificially reports a high volume in the north or south approaches. A designer applying this analysis may then design a longer ϕ_1. This policy is simulated in Fig. 5 and shows how sensitive such a system is to perturbations.

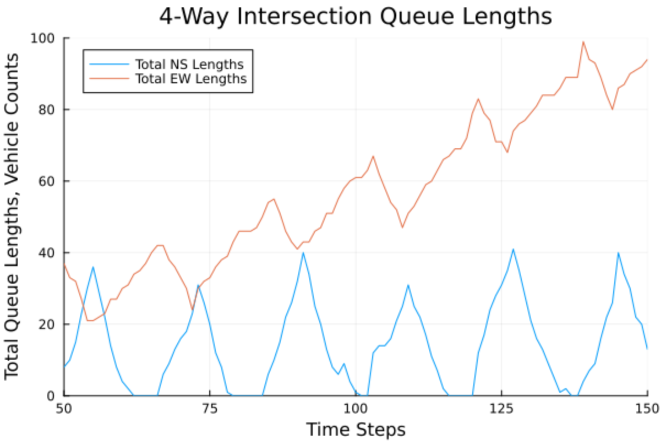

Fig. 5. When signal timing is perturbed, the intersection becomes oversaturated and unstable. In this simulation, the green phase of NS is increased 20% and the EW phase is decreased 25%. The EW approaches become unstable and cannot recover. After 1000 time steps, the total queue is $>$ 500 vehicles (an all-too common problem in the DMV area).

TABLE I
SIMULATED TRAFFIC FLOW CONDITIONS AND PHASE DESIGNATION

Approach (Dir.)	Sat. flow, s_i	App. flow, $q_i \sim \mu_i$	Green phase
N	5	2	ϕ_1
S	5	2	ϕ_1
E	3	1	ϕ_2
W	3	1	ϕ_2

V. MAXIMUM LIKELIHOOD REINFORCEMENT LEARNING

In this section, the background will be built to learn a signaling policy from the MDP. The goal is that the policy will instead reason over the states of the system, rather than relying on *a priori* estimates of the flow, to construct the signaling policy. In this work, the controller would detect the number of vehicles in the various queues and decide which signal to deploy by maximizing the expected value of being in a state.

A. High-level Construction

This work deploys maximum likelihood reinforcement learning on the MDP [8]. To start, the value associated with a state and action under a policy must be defined. Consider a state-action value function, $Q(s, a)$, that maps a state to the expected value of being in such a state:

$$Q(s, a) = R(s, a) + \gamma \sum_{s'} T(s'|s, a)V^{\pi}(s') \tag{6}$$

where π is the policy. The optimal policy at each state is then:

$$\pi^*(s) = \max_a Q(s, a) \tag{7}$$

While we may be able to reason over this as designers with exact knowledge of $T(s'|s, a)$ and $R(s, a)$, a much more useful formulation is if these are unknown. By assuming there is little or no knowledge of the system dynamics, a solution can then be generalized to autonomously learn signaling policies where the flows are unknown, which is a much more realistic situation. One applicable algorithm is maximum likelihood reinforcement learning.

Maximum likelihood reinforcement learning is a model-based method that constructs estimates of $T(s'|s, a)$ and $R(s, a)$ by running simulations of the environment and maintaining lists of state-action occurrences [8].

$$T(s'|s, a) \approx N(s, a, s')/N(s, a) \tag{8}$$

$$R(s, a) \approx \rho(s, a)/N(s, a) \tag{9}$$

where $N(s, a, s')$ is a count of occurrences of the (s, a, s') tuple and $\rho(s, a)$ is the running sum of rewards received at the (s, a) tuple. Continued iterations through the environment builds better estimates of these counts, where (6) can be accurately constructed to determine an optimal policy at a state, s.

B. Algorithm and Implementation

While these models rely on continued iteration through the state-space, not all the states are necessary to visit, especially in systems with prohibitively large state-spaces. Therefore, the balance between exploring new state-actions and better estimating visited ones is laid out in the next section. Exact implementation details are not discussed here (the reader is referred to [8]), but the general exploration strategy is described.

In each training epoch (300 total), 500 steps are taken. For the first 60 epochs, a heuristic policy is used to select actions. In this heuristic, the signal is switched when a queue clears (e.g. if N/S lanes are close to zero, then a = :EW). This builds initial counts of $N(s, a, s')$ and $\rho(s, a)$. Afterwards, a greedy-ϵ policy is deployed, where $\epsilon = 0.1$ to encourage some exploration of the space while primarily considering high-value state-action pairs. This careful trade-off of seeding $N(s, a, s')$ and $\rho(s, a)$ with a heuristic while then transition to an exploration/exploitation strategy yielded superior training results compared to just a greedy-ϵ policy. The psuedocode is laid out in Algorithms (1) and (2).

VI. Results and Discussion

The algorithm in sec. V was then implemented with some minor changes to the state-space. Consider, first, the state-space as written. In each approach (northbound, south, east, and west), any number of vehicles may be queued and a discrete state-space will quickly explode. However, this model approximation technique requires a finite state-space, so the depth of each lane is first limited to n vehicles. Before implementation, an analysis of the order of $Q(s, a)$ shows:

$$Q(s, a) \sim O(|n|^{\phi}|A|) \tag{10}$$

where n is the limited depth of a queue. For a four-phase approach with a limit of just 50 vehicles, Q(s, a) is 12.5M values! Not only will such a large Q(s, a) take very long to train, deployment in actual traffic problems may be computationally prohibitive. In this problem, the independent queues of

Algorithm 1 Max. Likelihood Algorithm

```
Initialize: MDP (env)
$N(s, a) = 0$          $\forall s, a$
$\rho(s, a) = 0$          $\forall s, a$
$N_p(s', s, a) = 0$    $\forall s, a$

# Perform for each episode
while $i \leq N_{eps}$ do
  # Update the estimates
  $T(s'|s, a) \leftarrow N_p(s', s, a)/N(s, a)$
  $R(s, a) \leftarrow \rho(s, a)/N(s, a)$

  # Value iteration with new estimates
  while $|V'(s) - V(s)|> 0.001$ do
    $V'(s) \leftarrow \max_a[R(s, a) + \gamma T(s, a)V(s)]$
  end while
  $Q(s, a) \leftarrow R(s, a) + \gamma T(s'|s, a)V(s)$

  # Step through the environment
  while j $<$ 500 do
    $a \leftarrow \pi(s, i)$
    $r, s'$ = act!(Env, a)
    $N_p(s', s, a) \leftarrow N_p(s', s, a) + 1$
    $N(s, a) \leftarrow N(s, a) + 1$
    $\rho(s, a) \leftarrow \rho(s, a) + r$
    $s \leftarrow s'$
    $j \leftarrow j + 1$
  end while
end while
Return $Q(s, a)$
```

Algorithm 2 Policy, $\pi(s, i)$

```
$\epsilon = 0.1$
if $i \leq \lfloor 0.2N_{eps} \rfloor$ then
  # Heuristic Policy
  if s[1] $<$ 5 then
    Return :EW
  else if s[2] $<$ 5 then
    Return :NW
  else
    Return s[3]
  end if

else
  # Greedy-$\epsilon$ Policy
  if rand([0,1]) $< \epsilon$ then
    Return rand($A$)
  else
    Return $\max_a[Q(s, a)]$
  end if
end if
```

lane groups during a phase need not be considered separately. By grouping all lane groups under a lumped queue, the order is now $O(|n|^2|A|)$ which is a much more reasonable 5000 state-actions to reason over when $n = 50$.

As per sec. V.B, 300 training epochs with 500 steps, each, are run. The heuristic is deployed for the first 60 epochs to construct the initial counts. After each training epoch, the Q(s, a) policy is evaluated with (7). In this evaluation, 5 episodes of 100 steps, each, are taken, and the total undiscounted reward is summed. The standard deviation amongst these 5 sums are computed. Fig. 6 plots this learning curve.

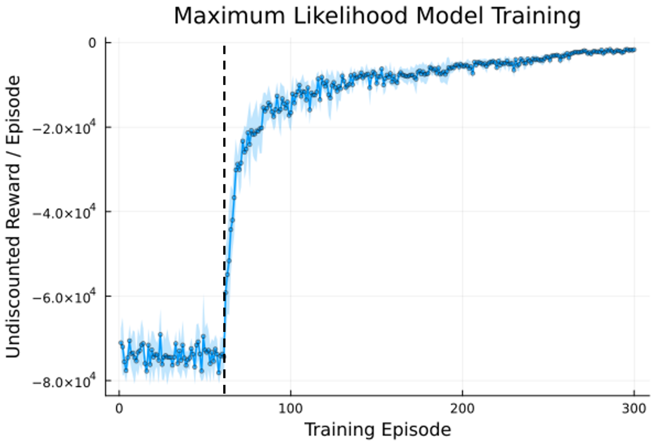

Fig. 6. Maximum likelihood reinforcement learning model training, evaluated with a = argmax[Q(s, a)]. The dotted line represents the end of the heuristic initialization and the start of a greedy-ϵ policy. The ribbon shows ± one standard deviation. The training is stable and consistently increases.

The training shows a marked improvement as the RL algorithm begins using the Q(s,a) estimate to guide exploration of the state-space. Among the 5 training episodes, the variance is low and the improvement from learning Q(s,a) is unlikely to be just sample-to-sample variance. Learning begins tapering after approx. 250 episodes, but the undiscounted reward is clearly maximized as estimates of Q(s,a) are improved and a better policy is constructed. To visualize the improvements of this training, an episode with a policy from (7) is plotted at 25, 50, 75, and 90% through the training (all found in the appendix). Early in the training, the queues are consistently at the 50 vehicle upper limit, as the signal system cannot clear the vehicles. However, as training improves, the signal is able to consistently keep the queues away from the saturation limit and it begins minimizing the queue lengths.

A. Head-to-Head Comparison

The learned policy of (7), with Q(s, a) from the final result of maximum likelihood training, is then compared head-to-head with a state-of-the-art method described in sec. IV.B. this method is deployed on the modified model for computational efficiency, but the results from sec. IV.B are still valid without loss of generality. Using the policy $\pi(s) = \text{argmax}_a Q(s, a)$ and the final trained Q(s, a), the signal system is simulated for 500 steps and the undiscounted reward is calculated. This is repeated for 50 episodes and the mean and standard deviation are computed in table II.

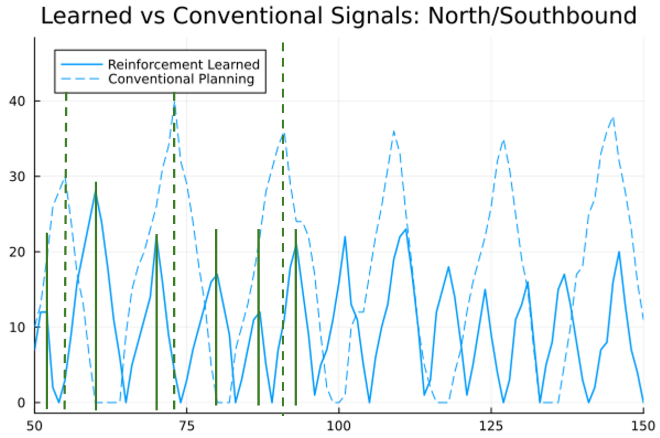

Fig. 7. NS lumped approach queue lengths on a simulated cycle. The dotted lines relate to the capacity and movement analysis, while the solid lines are the learned policy. The green, vertical lines show when the signal is green for the NS phase.

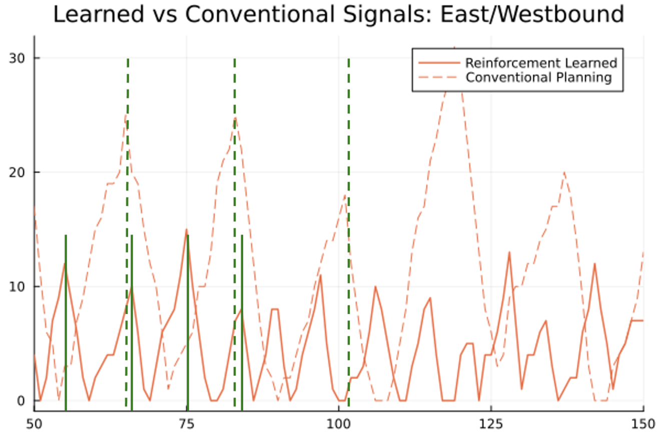

Fig. 8. EW lumped approach queue lengths on a simulated cycle. The dotted lines relate to the capacity and movement analysis, while the solid lines are the learned policy. The green, vertical lines show when the signal is green for the EW phase.

TABLE II
HEAD-TO-HEAD COMPARATIVE RESULTS

Method	Mean Score	Std. Dev
Capacity Analysis	-13,256	814.9
Learned Q(s, a) Policy	-9,822	1,244

The learned policy outperformed the state-of-the-art! By inspection of Fig. 8 and Fig. 7, the policy minimizes the peaks by essentially transitioning the light more frequently. Even though transitioning the light adds vehicles to every approach from the modelled all-red phase, which may be considered a detriment to a naive controller, the policy maintains low queues. Part of the success should be attributed to the controller's ability to reason over the states of the intersection, which a typical capacity analysis controller does not have access to.

B. Robustness

Consider an unusual traffic circumstance where traffic flows change substantially. Because the learned system reasons over the states, one potential advantage is that the learned controller may be robust to such circumstances. As a test, a new MDP is defined where, using the same construct as throughout, the NS approach flows are decreased to u_i = 1.5 and the EW are increased to u_i = 1.5. Using the $Q(s, a)$ trained on the *original* flow rates in table 1, the RL policy is evaluated head-to-head against the critical analysis derived from the original flow rates as well. The NS and EW queue lengths are plotted in 9 and 10 and clearly show the RL policy better handles the perturbed approach flows in the EW phase while performing similar to the critical analysis in the NS phase. For 50 simulated episodes, the mean undiscounted reward is -16,159 ± 3,341 under the RL policy, while the undiscounted reward under the critical capacity analysis is -24,413 ± 4,591. This RL policy is much more robust against changes to flow rate compared to the critical capacity analysis.

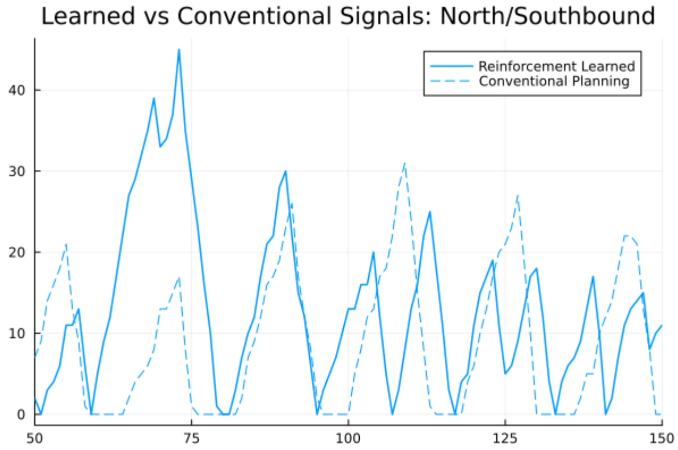

Fig. 9. NS lumped approach queue lengths on a simulated cycle with a **25% smaller** approach flow than trained on. The dotted lines relate to the capacity and movement analysis, while the solid lines are the learned policy. The conventional and and learned policies perform similarly.

VII. Future Work

The obvious limitation, though, is the explosion of the state-space. Attempting to model more complex phases (including turns and different flows in a single lane group) will quickly become computationally cumbersome to work with. Further, this work assumed that the states were known exactly, but modern intersections only get estimates of vehicles in a queue. The following are potential avenues for future work to resolve some of these issues:

- An algorithm that reasons over a continuous state-space may be more applicable with more complicated phasing. By letting the states (i.e. the queue lengths of the different phases) be continuous, the problem avoids computational overflow. A *very* important consideration, however, is that the system must remain certifiable. There can never be a way such that two groups occupy a the same position within an intersection at any given time. In discrete space, this was easier to enforce via implementation, but when working in continuous-space, this must be approached with caution.

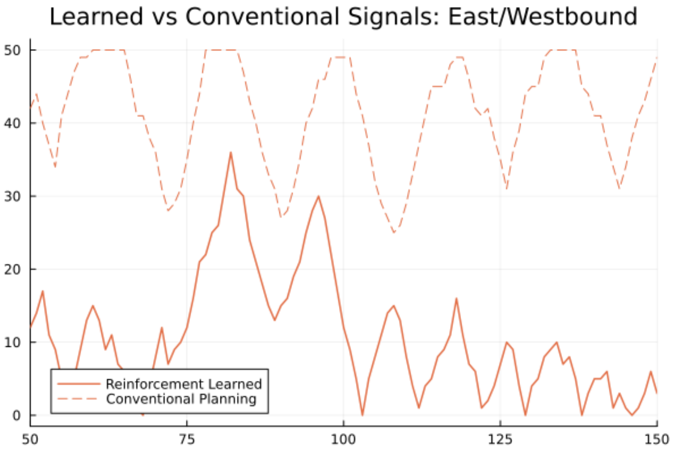

Fig. 10. EW lumped approach queue lengths on a simulated cycle with a **50% larger** approach flow than trained on. The dotted lines relate to the capacity and movement analysis, while the solid lines are the learned policy. The learned policy handles the perturbation much better than the conventional analysis.

- A POMDP model may find application in this problem by reasoning over estimates or observations of the states. In practice, loop sensors, camera detection, or other methods of vehicle counting may operate with some uncertainty. This was obviously not addressed in this work, but in practice, there will be uncertainty or potential for failure within these sensing system. A POMDP formulation and solution may address this, but again, the researcher should exercise caution in ensuring system safety certifiably.

VIII. Conclusions

This work demonstrates the application of a reinforcement-learning algorithm to a signalized intersection. An MDP is first constructed from a typical intersection, then a maximum likelihood reinforcement learning algorithm solves the MDP for an optimal policy. This learned policy significantly out-performs a conventional critical capacity analysis typically used by designers and appears robust against perturbation in the approach flow. Ultimately, by allowing the solver to reason over the states of the system, it switches the lights more frequently and judiciously to minimize queue lengths. While this work demonstrates success on a smaller construction, intersections in practice typically have multiple phases, where this implementation would become too computationally complex. Therefore future work may investigate continuous state-spaces or model reduction techniques.

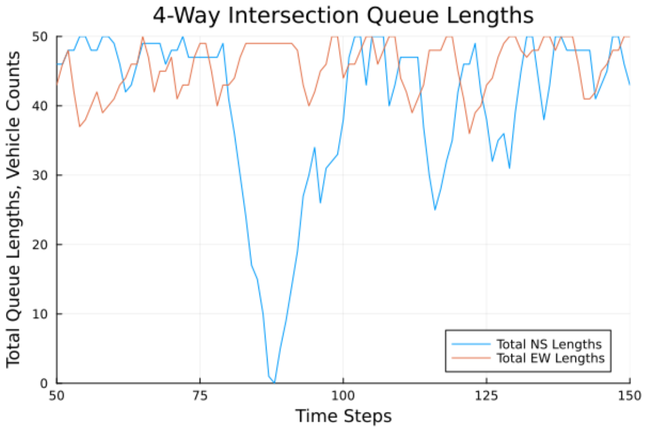

Fig. 11. Training episode following an a = argmax[Q(s, a)] policy with 25% of training complete.

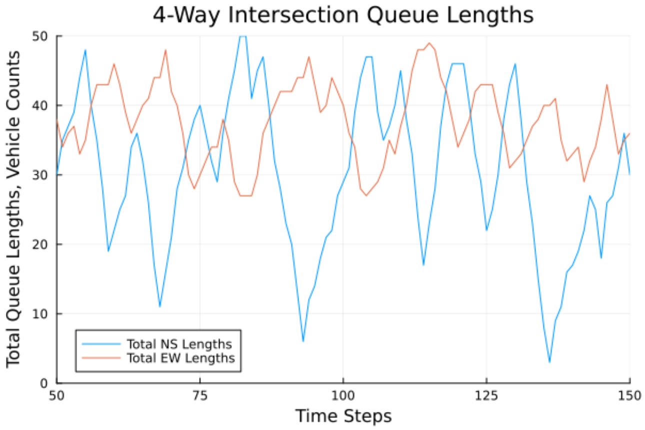

Fig. 12. Training episode following an a = argmax[Q(s, a)] policy with 50% of training complete.

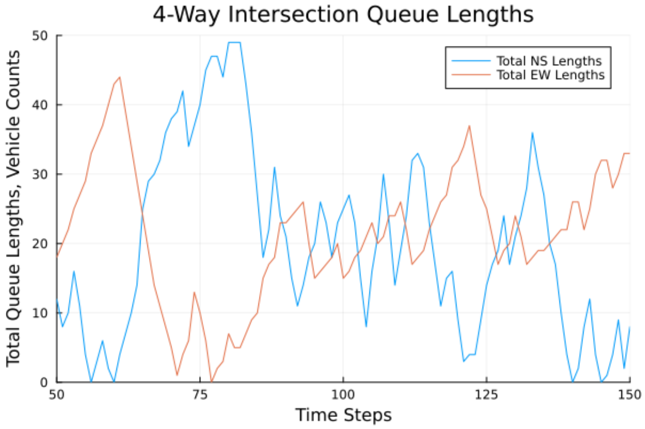

Fig. 13. Training episode following an a = argmax[Q(s, a)] policy with 75% of training complete.

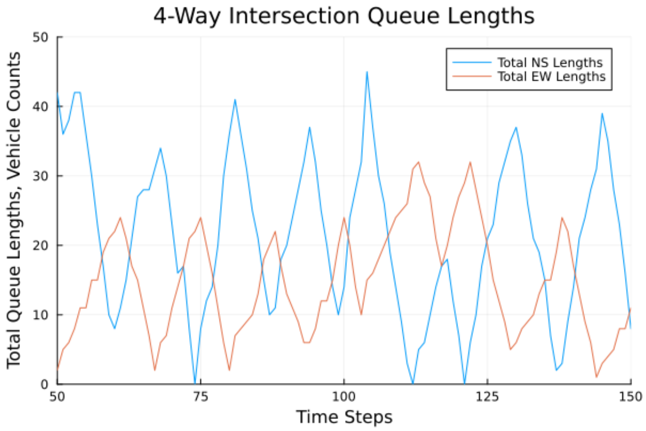

Fig. 14. Training episode following an a = argmax[Q(s, a)] policy with 90% of training complete.

IX. Appendix

References

[1] ”Traffic Signal Timing Manual,” US Dept of Transportation, Federal Highway Administration, Washington, DC, 2008.

[2] ”Traffic Detector Handbook,” US Dept of Transportation, Federal Highway Administration, Washington, DC. , Oct, 2006.

[3] ”Traffic Signal Design - Detection. TDOT Traffic Design Manual,” TDOT, Nashville, TN., 2016.

[4] B. T.-S.-H. Associates, ”The Benefits of Retiming/Rephasing Traffic Signals in the Back Bay,” Boston Transportation Dept, Boston, MA, 2010.

[5] L. B. Y. T. Chin, ”Exploring Q-Learning Optimization in Traffic Timing,” in Third International Conference on Computational Intelligence, Communication Systems and Networks, 2011.

[6] W. L. Han, ”Leveraging reinforcement learning for dynamic traffic control: A survey and challenges for field implementation,” Communications in Transportation Research, vol. 3, 2023.

[7] N. Garber, L. Hoel, ”Traffic and Highway Engineering,” 4th edition, Cengage Learning (2009).

[8] M. Kochenderfer, T. Wheeler, K. Wray, ”Algorithms for Decision Making,” The MIT Press, Cambridge, Massachusetts, (2022).