Reinforcement Learning For Adaptive Traffic Signal Control

Final Project, CS 229 (Machine Learning), Stanford University
Jeffrey Glick (jdglick@stanford.edu), M.S. Candidate (Management Science & Engineering)
11 December 2015

Introduction

By 2050, two-thirds of the world's 9.6 billion people will live in urban areas [2]. In many cities, opportunities to expand urban road networks are limited, so existing roads will need to more efficiently accommodate higher volumes of traffic. Consequently, there is a pressing need for technologically viable, low-cost solutions that can work with existing infrastructure to help alleviate increasing traffic congestion.

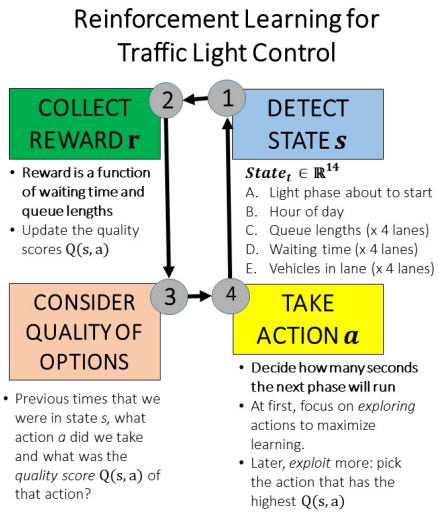

FIGURE 1: RL for Adaptive Traffic Signal Control

Improving the efficiency of traffic lights is key to reducing traffic congestion. Most static phase timings are chosen to work acceptably well in all conditions but are rarely optimal. Some modern traffic light control systems (often involving induction loops installed on road surfaces) utilize actuated logic. Such systems allow basic *if-then* rules to be pre-programmed by a trained traffic engineer with an understanding of local traffic dynamics. However, the installation, programming and maintenance of such systems is costly and time-consuming. Furthermore, the inflexible logic in these systems cannot adapt to changing environments and optimize control policies. The motivation of this study is to design, simulate and analyze an adaptive traffic signal control system that can out-perform the legacy traffic light control systems described above. This paper details efforts to to dynamically optimize a standard four-way intersection stop light utilizing reinforcement learning (specifically, the Q-learning algorithm).

The decision process is outlined in Figure 1. Every time the traffic light is about to enter a new phase, the control agent must first detect the state of the environment (eg. queue lengths). With this perception of the state, the agent must select some action. Specifically, the agent decides how many seconds the upcoming phase will run for. At the next decision point, depending on the *quality* of new state (where quality is defined quantitative by an objective function), the control agent is either punished or rewarded. The control agent initially explores a wide range of states and actions. Eventually the agent learns the best action for any given state and picks the best action with a high likelihood.

Related Work

There is growing body of work in which employ model-less temporal difference reinforcement learning methods like Q-learning to a range of traffic optimization problems. Two relatively early studies by Weiring [45, 47] provided a framework for simulating traffic systems using micro-simulation software and comparing performance of different reinforcement learning approaches. Mersetivc's 2014 simulation study showed that Q-learning outperforms static and actuauted logic control systems [28]. However, instead of using realistic, temporally dynamic traffic conditions, his control agent was trained in relatively static environments in which the traffic was either 'saturated' and 'unsaturated'. Steingrover explores the feasibility of deploying RL-based traffic light control policies in a sub-network of 16 intersections in which decision-making is optimized locally, but a penalty term is utilized for queue spill-over that affects an adjacent intersection [38]. Q-learning performs well when each local control agent is provided with full, perfect information about the state of the entire sub-network and neighboring decision making processes. Several other recent studies [14, 5, 24, 21, 30] all create simulation test beds to investigate whether Q-learning can be applied to traffic control problems; all of them yield promsing academic results, but the prospects of deploying these system designs into the real world is uncertain.

A recent literature in this field is provided by Mannion [26]. He considers the feasibility of deploying some RL techniques outside of simulation test beds and into the real world. His discussion raises the following points: First, simulated arrival rates and traffic dynamics into simulation test beds often lack the volatility and chaos of normal traffic flows. Second, he questions the robustness of the learning pipelines and processes researched. Many of the proposed systems would require significant tuning and supervision to deal with unique nuances of specific intersections.

Finally, he notes that many researchers make the implicit assumption that the learning agent has perfect, real-time information about the world around it. Providing the learning agent access to a wide feature set is easily done in a simulation environment and certainly useful for comparing performance of different algorithms. In the context of the RL-cycle (Figure 1), efficient learning agents with access to full information can precisely associate each state-action-state move some

perceived value. However, when considering the feasibility of deploying an intelligent and adaptive traffic control system, providing the agent access to perfect information would require extensive vehicle-to-control agent communication in conjuction with some combination of deployed hardware systems including cameras, radar, induction loops, counters or other physical instrumentation. These systems are expensive to install, calibrate, operate and maintain.

Garcia et. al. provides an thorough overview of implementation challenges associated with *temporal difference methods* like *Q-learning* [16] with a focus on how to map an infinite continuous state spaces to something more manageable. The learning agent needs a framework to generalize its experiences so that it can tractably approximate value functions. The study suggest that discretizing the state space with vector quantization (i.e. *K-means clustering algorithm*) provides a reliable approach for mapping similar continuous states into discrete buckets. We borrows elements of that discretization approach herein.

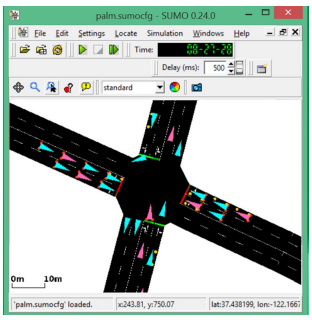

FIGURE 2: SUMO

Simulation Setup, Data Generation & State Space Definition

This study utilizes a multi-agent micro-simulation tool to simulate a traffic system involving cars, roads, intersections and traffic lights. Open-source software `SUMO` (*Simulation of Urban MObility*) was selected for the study. The software is highly flexible, well documented and supports control of traffic light signals using the API calls from an external `Python` script using the Traffic Control Interface (`TraCI`) package. It has extensive command-land functionality and a graphic user interface. Several other researchers have used `SUMO` for similar studies including recent work out of Google Research [11, 9].

Phase Number	Description	Seconds
0	East/West LEFT	8
1	YELLOW Transition	4
2	East/West STRAIGHT	28
3	YELLOW Transition	4
4	North/South LEFT	8
5	YELLOW Transition	4
6	North/South STRAIGHT	28
7	YELLOW Transition	4

FIGURE 3: Fixed phase times

The single intersection of Palm and Arboretum on Stanford Campus was selected[1]. A map of the intersection was first exported from `OpenStreetMap.org`. Next, the `map.osm` file was edited and cleaned in the *Java Open Street Map Editor*.

Among other tasks, entry points for vehicle spawning were identified, speed limits were set, road edges were properly connected at valid junctions and the traffic control signal was configured[2]. The intersection was purposefully modified from its true form, using two lanes on all approaches into the intersection and using a common eight-phase stop light cycle. The `map` file was then converted into an `.xml` file which can be read by the `SUMO` configuration file.

Next, we had to generate arrivals data for the simulation. Based on some observations during both rush-hour and off-peak travel times, synthetic data was generated to approximate non-homogeneous arrival rates into the system. Additionally, the decision of a car to (1) turn left (2) go straight or (3) turn right for each of the four approaches was approximated, considering observed behavior of the intersection. Polynomial functions were fit to this synthetic data in order estimate time-dependent arrival rates of cars arriving at one of the four spawn points and exiting out of one of the three other edges (Figure 4).

With the assumption that vehicles arrive according to a Poisson process with the mean arrival rate determined by the fitted functions (and some added randomness), a schedule of arrivals was randomly generated in a `routes.xml` file for every 24-hour run of the simulation. The file provides a start location, start time and end location for vehicles over the entire 24-hour simulation. The vehicle will automatically choose the proper lane depending on if it is turning left, right and will pick either lane if it is going straight. Vehicles may turn left during cycles 2 and 6 (Figure 3)) but do not have the right-away. Right turns on red are allowed and all basic rules and courtesies are followed. Realistic acceleration, deceleration lane maneuvering and other behaviors are also programmed into the simulation.

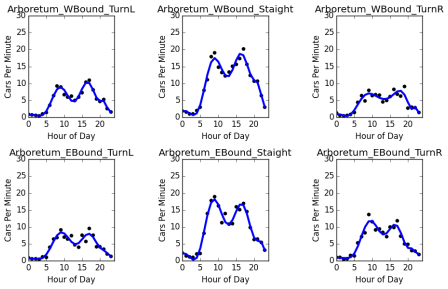

FIGURE 4: Variable arrival and turn rates

Motivated by a desire to design a system which requires limited (or no) investment in locally deployed physical sensors, this research seeks to design an end-to-end learning pipeline and control system which uses *limited* input information. We consider the use of data which might realistically be available from smart phone geo-location data (being emitted by passengers in cars traversing the intersection).

So, where i denotes each of the four intersection approach lanes lane ($i \in \{North, South, East, West\}$), the state of the system $S \in \mathrm{R}^{14}$ is defined by the following elements - k: phase which is about to start; h: hour of the day[3]; q_i: the queue

[1]The author bikes through this intersection daily and is often frustrated by its sub-optimal fixed phase timings!

[2]A baseline fixed-phase light timing shown in (Figure 3)) was selected because on the fact that it minimized the average objective function compared to other reasonable fixed timing policies

[3]By including the h as an element in our state s, we make the inherent assumption that traffic dynamics have cyclic temporal dynamics.

size of each of the four approaches [4]; w_i: current cumulative waiting time of all of the stopped cars [5]; nv_i: total number of vehicles in each lane within a fixed radius around the intersection.[6]

Methods & Experimentation

Q-learning is an off-policy, temporal difference (TD) control algorithm[7] originally developed by Watkins in 1989 [43]. By exploring states, actions and rewards, the intent is to learn *quality values* $Q(s, a)$ for each state and action which serve to estimate the true optimal value function. The algorithm continuously updates these $Q(s, a)$ values in an asynchronous manner (a mechanism for delayed reinforcement) [18]. The conditions for eventual convergence to an optimal $Q^*(s, a)$ is that each pair (s, a) be visited an infinte number of times and the learning rate α be updated in the manner shown in equation (2) below [42]. The one step update of the *Q-learning* algorithm is:

$$Q(s,a) := (1-\alpha(s,a))Q(s,a) + \alpha(s,a)\Big[r + \gamma \text{max}_{a' \in A} Q(s',a')\Big] \tag{1}$$

As the first step in our Q-learning pipeline, we need to map our R^{14} state space to a set of discrete states. We employ vector quantizer $VQ_{h,k}$ (i.e. k-means clustering algorithm) to achieve this[16]. For each hour of the day h and each of the four light phases where we can take an action k, our quantizer maps the states as follows:

$$VQ_{h,k}(S) : S \rightarrow C_{h,k} \ \forall \ h, k$$

where $C_{h,k}$ is our full set of centroids for each unique h and k. We next consider the question of how to decide $\mathbf{card}(C_{h,k})$, which is the budget of clusters allocated for each h, k pair. Following significant experimentation, we developed insight that $\mathbf{card}(C_{h,k})$ should be proportional to $\sum_{i=1}^{12} \mathbf{Var}(S_i)$. Why? If the continuous state space for a given h, k pair is highly variable (i.e. rush hours when traffic volume can spike suddenly) then a relatively larger number of clusters should be assigned in order to ensure that the $VQ_{h,k}$ can perform a sufficiently precise mapping of similar states. On the other hand, if the range states is relatively compact in terms of Euclidean distance (i.e. 2 AM), then a smaller number of clusters can be assigned and perform their intended task with the same level of precision (map similar states to the same discrete state). So, in order to choose $\mathbf{card}(C_{h,k})$ and generate a $VQ_{h,k}(S)$ for each h and k, we executed the following procedure:

For each h, k:

(1) Passively observe the system for 3 days. Record all available state vectors $\in \mathrm{R}^{12}$ and concatenate the n observations into a matrix $\mathbf{S}$ where each column j represents a dimension of the state.

(2) Let $\mathbf{Stdev}(S)_j$ be the standard deviation for each column j of $\mathbf{S}$.

(3) Let $\mathbf{card}(C_{h,k}) = \sum_j \mathbf{Stdev}(S)_j$.[8]

(4) Run the k-means clustering algorithm on $\mathbf{S}$ using the selected number of centroids; store the resulting $VQ_{h,k}(S)$ function for later use to map state vectors S to some $C_{h,k}$

(5) Given the existence in some (h, k) a future state s will be mapped to a discrete state according to $VQ(s) = \text{argmin}_{y \in C}\{dist(s, y)\}$ where *dist* is the Euclidean distance.

Next, the learning rate $\alpha(s, a)$ used in equation (1) above is defined as:

$$\alpha(s,a) = \frac{1}{n(s,a)} = 1, \frac{1}{2}, \frac{1}{3}, \frac{1}{4}, ... \tag{2}$$

where $n(s, a)$ is the number of times that action a has been taken from state s. $\alpha(s, a)$ controls how we weigh the most recent experience and the past experiencing when updating a quality value $Q(s, a)$. Early in the learning process when $\alpha(s, a) = 1$, the initialized value of $Q(s, a)$ is completely wiped out and replaced by the RHS of the update (ref equation (1)). As $n(s, a)$ increases, α shrinks and we place relatively less weight on a single new experience in the $Q(s, a,)$ update process.

The discount factor γ is some value $\in [0, 1]$. In the context of a vehicle control problem, we want to set γ close to one in order to prevent myopic decision making (a value of $\gamma = 0.9$ was found to work well).

A wide range of punishment (a.k.a reward) functions have been used in the literature, with the sum of squared queue sizes being a popular option. We compute an objective function that incorporates both queue sizes and cumulative waiting time of stopped cars in the queue. While these two performance indicators tend to be highly (positively) correlated, we do

[4] queue sizes is the count of stopped cars waiting at the intersection; this is the most important metric from a traffic-optimization standpoint and features prominently in the reward function of related research studies

[5] waiting time is the most important metric from the passengers point of view

[6] nv provides the control agent some awareness of number of cars about to arrive to the intersection.

[7] Off-policy algorithms can update the estiamted value functions using hypothetical actions where an assumption is made that the best action will be taken; on-policy algorithms only make updates based strictly on experiences[40]

[8] $\mathbf{card}(C_{h,k})$ is the number of clusters and corresponds to the the `nclusters` argument in the `sklearn.cluster.KMeans(·)` function in `Python`)

want to have some trade-off between the two in order to avoid situations where a single car might be waiting in a 1-car queue for an infinite amount of time. Our reward function is:

$$r = -\sum_{i=1}^{4} \beta_q (q_i)^{\theta_q} + \beta_w (w_i)^{\theta_w} \tag{3}$$

Selections of of both the leading coefficients and power terms in this function have significant impact on the the eventual policies arrived at. Based on experimentation and observation, we utilized $\beta_q = 1$ and $\beta_w = 2$ in order to bring q_i and w_i onto approximately the same scale. Using a quadratic term on q is a common in other studies, which encourages a balancing of queue lengths. [5] Both θ_q and θ_w were set to 1.5.

We next define the discrete set of actions $a_k \in A_k$ where k denotes the light phase where some decision on an action needs to be made. $k \in 0, 2, 4, 6$ since the odd numbered k values are the yellow transition phases which are fixed at 4 seconds each. We choose $\mathbf{card}(A_k) = 8 \ \forall \ k$ and assigned each $a_k^{(i)}$ as a discrete integer value between reasonably designated lower and upper constraint on the time that each phase k can last. For $k \in 0, 4$, we define $A_k = \{3, 4, 5, 6, 7, 8, 9, 10\}$ (seconds) and for $k \in 2, 6$ we define $A_k = \{10, 14, 18, 22, 26, 30, 34, 38\}$ (seconds).

ϵ is a control parameter which manages the *explore* vs. *exploit* objectives of the Q-learining process. ϵ is typically fixed at some value, usually between 0.8 and 1. For early stages of the learning process, an '$\epsilon - soft$' policy is utilized where the best action (with the highest $Q(s, a)$ value is chosen with probability $1 - \epsilon$ and the remaining actions are chosen with uniform probability. After some reasonable threshold of experience ($\sum n(s, a)$) is gained, a control policy will then be shifted to '$\epsilon - greedy$'. Under this control policy, the action with the highest $Q(s, a)$ is chosen with probability ϵ. The remaining probability mass is spread over the other options uniformly; consequently, they are only visited occasionally.

As an alternative to the tradition ϵ-greedy or ϵ-soft control policies, this study utilizes *softmax* approach by which $\epsilon(s)$ is a function of the state s.

$$\epsilon(s) = \frac{\frac{1}{\mathbf{card}(A)} \left| \sum_{a \in A} Q(s, a) \right|}{\left(\frac{1}{\mathbf{card}(A)} \sum_{a \in A} n(s, a) \right)^{\nu}} \tag{4}$$

Due to our highly varied values of $Q(s, a)$ (values as low as -1200 at times), this approach was initially utilized to prevent numerical underflow but there are other advantages. It also dynamically updates as $n(s, a)$ grows, codifying a gradual shift towards a greedier control policy. ν can be adjusted as a tuning parameter to gradually control the transition from an exploratory to exploitative policy. This study utilized $\nu = 1$.

This definition of $\epsilon(s)$ then serves as the key input to assign probability mass across the set of actions for a given state:

$$P_s(a) = \frac{\exp[Q(s, a)/\epsilon(s)]}{\sum_{a \in A} \exp[Q(s, a)/\epsilon(s)]} \ \forall \ a \in A \tag{5}$$

Note that this function guarantees that $\sum_a P_s(a) = 1$ and actions with a higher $Q(s, a)$ will be assigned a relatively higher probability mass. When it comes time to select an action, random number generation is used to select an action based on this discrete probability distribution.

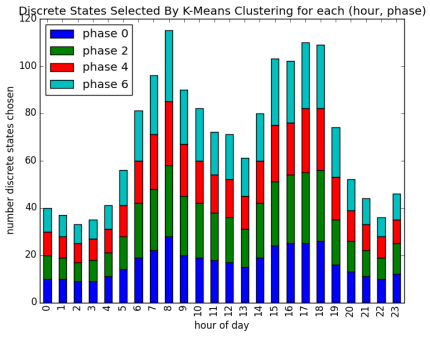

FIGURE 5: Results of unsupervised state space discretization via k-means

Results & Analysis

Our learning pipeline begins with an observation period of a 3 days. Our agent uses k-means clustering in order to discretize the state within each (h, k) pair. The number of centroids chosen is shown in Figure 5. Since variance of the state vector values are higher during rush hours, a relatively higher proportion of centroid clusters were utilized - exactly as expected. In total, the procedure produced $N = 1694$ unique states. We found that this produced a set $VQ(\cdot)$ mapping functions with roughly equal precision. Furthermore, compared to initial naive attempts to discretize the state space uniformly, the distribution of $n(s, a)$ occurrences across the 1694 states was much less sparse, providing some indication that we are effectively utilizing the allocated discrete states (by the end of day 5, the average number of visits to each state was 12.6 with a standard deviation of 11.0; only 52 hadn't been visited a single time). We experimented with different values N (N = some hypothetical centroid budget) and the implications for the Q-learning process were not surprising: the greater the number of discrete states, the the longer exploratory period for the Q-learning algorithm, but the better the eventual performance. Under regime of a very large N, far more (s, a) pairs need to be visited to build reasonably accurate $Q(s, a)$ scores. With too small of an N, we improved our mean objective score faster but performance plateaued at a lower level.

Our primary metrics for assessing the performance of the algorithm on a day-over-day basis include the mean, median and min of the objective function. A 50-day simulation run shows the evolution of these summary statistics in Figure 6. The baseline values of each summary statistic (computed from the 3-day observation period of the static policy) are plotted as horizontal black lines for reference. We can see that the daily mean objective function at first oscillates around the baseline objective value and then after the second week gradually improves. At the end of day 50, the mean reaches -35, which a 40% improvement from the baseline performance. The daily median objective value also tracks upward and is about 40% higher after day 50. We also monitor the minimum objective function because we care about queue blow-ups and other exceptionally bad situations over the course of a day. The min daily value initially drops but starts to recover towards the baseline.

These results show that the Q-learning algorithm does indeed improving average system performance in terms of the objective function (a composite measure of queue length and waiting time), but at an snails' pace! We don't see any convergence of performance even after 50 days when each state has been visited an average of 147 times. We examine our discrete probability distributions for actions. At this point in the learning process, for some state s, we'd like to be well into a $\epsilon(s)-$*greedy* control policy, picking $\text{argmax}_{a \in A} Q(s,a)$ a vast majority of the time. However, at day 50, an average of 34% probability weight (median of 31% probability weight) was applied to the $\text{argmax}_{a \in A} Q(s,a)$. An average of 71% probability mass is assigned to the top three options. Consequently, we recognize that we should have utilized a more aggressive value for $\epsilon(s)$ and could have increased the value of ν to some value between 1 and 2 to attempt to speed up convergence to help get a greedier control policy at this late stage.

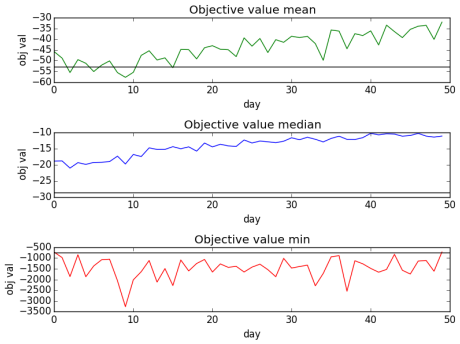

FIGURE 6: Mean, Median and Min of Objective Function after 50 Days

By initializing *all* of the $Q(s,a)$ pairs at be zero, we ensured that each would be visited at least once since the rewards are all negative. From a Bayesian standpoint, this was a a flat prior and it likely would have been helpful to initialize with something other than a flat prior to encourage most of the early exploration within a subset of actions that we believed would be best for some particular state.

As the algorithm progressed, a weighted moving average of the objective function was visualized to provide insight about temporal and day-over-day performance could be analyzed for any discernible patterns. Figure 7 provides a view about the relatively noisy nature of the algorithm during early days when the control policy encourages exploration. Later, we would see these moving average curves incrementally shift upward but with a good deal of stochasticity.

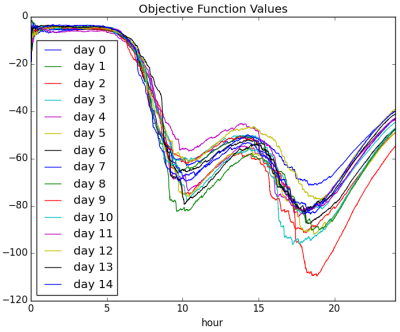

FIGURE 7: Moving average of objective function (first 15 days)

Conclusion & Future Work

This study was carried out to confirm the value of applying reinforcement learning to improve traffic intersection system performance. We carefully set up a complex multi-agent model with realistic traffic flows. We used an unsupervised method to reasonably partition an infinite state space and then tasked our learning agent to learn optimal policies in a in a temporally dynamic environment. This study provides another confirmation that Q-learning shows promise for solving traffic light control problems and improving overall system performance compared to fixed policies. However, in spite of the relatively straightforward nature of the algorithm, the lack of convergence and difficulties with selecting model parameters also reveals the non-trivial challenges of deploying a Q-learning learning pipeline.

With the theoretical benefits of Q-learning well-established, more work needs to move research out of simulation test beds and into the field. Much of the the industry activity attempting to exactly this are developing business models which call for the deployment of extensive physical instrumentation at intersections (including sensors, radar, cameras, induction loops and other equipment). Such systems show promise and are attractive from a revenue standpoint, but they are very expensive for the customer (i.e. a city government) to purchase and certainly not affordable for much of the developing world.

We suggest that researchers focus on the desining a system that uses limited information. We might imagine that an agent only has access to geo-location data for some sub-set of vehicles which is only near-real-time (i.e. a 2 or 3 second time lag). In such a regime, an agent would first need to learn how to generate estimates about what state the system is in and develop novel mechanisms for verifying these estimates based on subsequent signals. The output of that first learning problem would then serve as noisy inputs to the Q-learning traffic light control process studied above. Developing learning pipelines and designing systems which are low-cost, reasonably effective, deployable and require minimal hands-on support from experts should be the focus.

Acknowledgments

Michael Bennon, Managing Director at the Stanford Global Projects Center helped with problem formulation and has provided ongoing feedback. CS 229 Students Allen Huang and Jesiska Tandy contributed to the early brainstorming and literature review phases of this project.

References

[Pyt] *Python packages Numpy (polyfit), Scikitlearn (cluster.Kmeans).*

[2] (2015). World popluation prospects. Technical report, UN Department of Economic and Social Affairs Population Division.

[3] Abdelgawad, H., Rezaee, K., El-Tantawy, S., Abdulhai, B., and Abdulazim, T. (2015). Assessment of adaptive traffic signal control using hardware in the loop simulation. In *Intelligent Transportation Systems (ITSC), 2015 IEEE 18th International Conference on*, pages 1189–1195. IEEE.

[4] Abdoos, M., Mozayani, N., and Bazzan, A. L. (2013). Holonic multi-agent system for traffic signals control. *Engineering Applications of Artificial Intelligence*, 26(5):1575–1587.

[5] Abdulhai, B., Pringle, R., and Karakoulas, G. J. (2003). Reinforcement learning for true adaptive traffic signal control. *Journal of Transportation Engineering*, 129(3):278–285.

[6] Alizadeh, F. and Papp, D. (2013). Estimating arrival rate of nonhomogeneous poisson processes with semidefinite programming. *Annals of Operations Research*, 208(1):291–308.

[7] Araghi, S., Khosravi, A., and Creighton, D. (2015). A review on computational intelligence methods for controlling traffic signal timing. *Expert Systems with Applications*, 42(3):1538–1550.

[8] Asmuth, J. T. (2013). *Model-based Bayesian reinforcement learning with generalized priors.* PhD thesis, Rutgers University-Graduate School-New Brunswick.

[9] Baluja, S., Covell, M., and Sukthankar, R. (2015). Approximating the effects of installed traffic lights: A behaviorist approach based on travel tracks. In *IEEE Intelligent Transportation Systems Conference.*

[10] Brys, T., Pham, T. T., and Taylor, M. E. (2014). Distributed learning and multi-objectivity in traffic light control. *Connection Science*, 26(1):65–83.

[11] Covell, M., Baluja, S., and Sukthankar, R. (2015). Micro-auction-based traffic-light control: Responsive, local decision making. In *IEEE Intelligent Transportation Systems Conference (ITSC-2015).*

[12] de Oliveira, D., Bazzan, A. L., da Silva, B. C., Basso, E. W., Nunes, L., Rossetti, R., de Oliveira, E., da Silva, R., and Lamb, L. (2006). Reinforcement learning based control of traffic lights in non-stationary environments: A case study in a microscopic simulator. In *EUMAS.*

[13] El-Tantawy, S. and Abdulhai, B. (2012). Multi-agent reinforcement learning for integrated network of adaptive traffic signal controllers (marlin-atsc). In *Intelligent Transportation Systems (ITSC), 2012 15th International IEEE Conference on*, pages 319–326. IEEE.

[14] El-Tantawy, S., Abdulhai, B., and Abdelgawad, H. (2014). Design of reinforcement learning parameters for seamless application of adaptive traffic signal control. *Journal of Intelligent Transportation Systems*, 18(3):227–245.

[15] Fleck, J. L., Cassandras, C. G., and Geng, Y. (2015). Adaptive quasi-dynamic traffic light control.

[16] Garcıa, J., López-Bueno, I., Fernández, F., and Borrajo, D. (2010). A comparative study of discretization approaches for state space generalization in the keepaway soccer task. *Reinforcement Learning: Algorithms, Implementations and Aplications. Nova Science Publishers.*

[17] Guez, A., Silver, D., and Dayan, P. (2012). Efficient bayes-adaptive reinforcement learning using sample-based search. In *Advances in Neural Information Processing Systems*, pages 1025–1033.

[18] Humphrys, M. (1996). Action selection methods using reinforcement learning. *From Animals to Animats*, 4:135–144.

[19] Iša, J., Kooij, J., Koppejan, R., and Kuijer, L. (2006). Reinforcement learning of traffic light controllers adapting to accident. *Design and Organisation of Autonomous Systems*, pages 1–14.

[20] Jadhao, M. N. S. and Kulkarni, M. P. A. (2012). Reinforcement learning based for traffic signal monitoring and management. In *International Journal of Engineering Research and Technology*, volume 1. ESRSA Publications.

[21] Jin, J. and Ma, X. (2015). Adaptive group-based signal control by reinforcement learning. *Transportation Research Procedia*, 10:207–216.

[22] Kalganova, T., Russell, G., and Cumming, A. (1999). Multiple traffic signal control using a genetic algorithm. In *Artificial Neural Nets and Genetic Algorithms*, pages 220–228. Springer.

[23] Khamis, M. A. and Gomaa, W. (2014). Adaptive multi-objective reinforcement learning with hybrid exploration for traffic signal control based on cooperative multi-agent framework. *Engineering Applications of Artificial Intelligence*, 29:134–151.

[24] Koltovska, D. and Bombol, K. (2014). Intelligent agent based traffic signal control on isolated intersections. *TEM Journal*, 3(3):216–222.

[25] Lin, W.-H. and Wang, C. (2004). An enhanced 0-1 mixed-integer lp formulation for traffic signal control. *Intelligent Transportation Systems, IEEE Transactions on*, 5(4):238–245.

[26] Mannion, P., Duggan, J., and Howley, E. (2015a). An experimental review of reinforcement learning algorithms for adaptive traffic signal control. *Autonomic Road Transport Support Systems, Autonomic Systems. Birkhauser/Springer*.

[27] Mannion, P., Duggan, J., and Howley, E. (2015b). Learning traffic signal control with advice. In *Proceedings of the Adaptive and Learning Agents workshop (at AAMAS 2015)*.

[28] Marsetič, R., Šemrov, D., and Žura, M. (2014). Road artery traffic light optimization with use of the reinforcement learning. *PROMET-Traffic&Transportation*, 26(2):101–108.

[29] Massey, W. A., Parker, G. A., and Whitt, W. (1996). Estimating the parameters of a nonhomogeneous poisson process with linear rate. *Telecommunication Systems*, 5(2):361–388.

[30] Medina, J. and Benekohal, R. (2014). Sensitivity of reinforcement learning agents to aggregated sensor data in congested traffic networks. In *T&DI Congress 2014@ sPlanes, Trains, and Automobiles*, pages 719–726. ASCE.

[31] Medina, J. C. and Benekohal, R. F. (2012). Agent-based traffic management and reinforcement learning in congested intersection network. *NEXTRANS Project*, (072IY03).

[32] Moghaddam, M. J., Hosseini, M., and Safabakhsh, R. (2015). Traffic light control based on fuzzy q-leaming. In *Artificial Intelligence and Signal Processing (AISP), 2015 International Symposium on*, pages 124–128. IEEE.

[33] Ozan, C., Baskan, O., Haldenbilen, S., and Ceylan, H. (2015). A modified reinforcement learning algorithm for solving coordinated signalized networks. *Transportation Research Part C: Emerging Technologies*, 54:40–55.

[34] Rezzai, M., Dachry, W., Moutaouakkil, F., and Medromi, H. (2015). Designing an intelligent system for traffic management. *Journal of Communication and Computer*, 12:123–127.

[35] Robinson, A. and Christopher, M. (2009). Learning traffic light control policies.

[36] Sanchez, J. J., Galan, M., and Rubio, E. (2004). Genetic algorithms and cellular automata: A new architecture for traffic light cycles optimization. In *Evolutionary Computation, 2004. CEC2004. Congress on*, volume 2, pages 1668–1674. IEEE.

[37] Srinivasan, D., Choy, M. C., and Cheu, R. L. (2006). Neural networks for real-time traffic signal control. *Intelligent Transportation Systems, IEEE Transactions on*, 7(3):261–272.

[38] Steingrover, M., Schouten, R., Peelen, S., Nijhuis, E., and Bakker, B. (2005). Reinforcement learning of traffic light controllers adapting to traffic congestion. In *BNAIC*, pages 216–223. Citeseer.

[39] Suryavanshi, S. and Kotaru, M. (2013). Predicting gas usage as a function of driving behavior.

[40] Sutton, R. S. and Barton, A. G. (1998). *Reinforcement Learning: An Introduction*. The MIT Press.

[41] Wang, Y., Won, K. S., Hsu, D., and Lee, W. S. (2012). Monte carlo bayesian reinforcement learning. *arXiv preprint arXiv:1206.6449*.

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

[43] Watkins, C. J. C. H. (1989). *Learning from delayed rewards*. PhD thesis, University of Cambridge England.

[44] Weinberg, J., Brown, L. D., and Stroud, J. R. (2007). Bayesian forecasting of an inhomogeneous poisson process with applications to call center data. *Journal of the American Statistical Association*, 102(480):1185–1198.

[45] Wiering, M. et al. (2000). Multi-agent reinforcement learning for traffic light control. In *ICML*, pages 1151–1158.

[46] Wiering, M., Van Veenen, J., Vreeken, J., and Koopman, A. (2004a). Intelligent traffic light control. *Institute of Information and Computing Sciences. Utrecht University*.

[47] Wiering, M., Vreeken, J., Van Veenen, J., and Koopman, A. (2004b). Simulation and optimization of traffic in a city. In *Intelligent Vehicles Symposium, 2004 IEEE*, pages 453–458. IEEE.