Transfer Learning Acceleration for Reinforcement Learning of Traffic Light Control Policy

Reid Christopher, Woo Maeng, and Masafumi Endo

Abstract

A lot of time and money are being wasted on roads due to traffic congestion in the world. Many methods have been applied to optimize traffic flow, one of which being reinforcement learning methods to control traffic light signals. One issue with implementing reinforcement learning methods is that most approaches require large amounts of data for training and start with rather poor performance. If we are to ever learn using these methods in the real world, the starting performance must have a reasonable lower bound and the amount of data used to learn should be decreased. In this paper, we aim to implement Reinforcement Learning with Transfer Learning to generate a traffic light control system for a 4-way intersection. The model is first trained to mimic a typical traffic light control system which is based on a fixed time sequence of phases. Then, the learned features are transferred to evolutionary and actor critic learning methods to improve from this base line performance. We show that this transferring of policy improves initial performance for both algorithms, but only provides stable improvement for the actor-critic method, while the evolutionary algorithm fails to improve from its transferred policy.

I. Introduction

The first modern electric traffic light was introduced in early 20th century and was operated manually by human from a tower in the street [1]. It was developed due to the chaos on roads caused by the disorder of cars, bicycles, horses, wagons and pedestrians without any regulations. While there are no horses or wagons on roads anymore, the number of automobiles on roads has dramatically increased. Traffic light systems have helped to alleviate this disorder on roads[2], but people still spend a lot of unnecessary time on roads waiting for green lights. It was estimated that congestion caused urban residents to travel an extra 6.9 billion hours and consequently use an additional 3.1 billions of gallons of fuel in United States in 2014 [3]. Also, traffic congestion cost the British, French, German and American economies over $200 billion in 2013 [3]. One of the effective ways to ease traffic congestion is expanding transportation capacity and building higher quality roads to deal with higher traffic flows. However, this solution requires a significant amount of money and time, and causes further problems with the traffic that is attempting to be improved. Hence, using existing road and improving traffic light adaptability is an appealing alternative.

One of the most popular traffic light control system currently being used are fixed sequence policy. Such traffic lights usually change phase using a static, predefined policy based purely on time. This strategy can cause increases in travel time since the static policy does not reflect the dynamic situation of traffic flow, such as congestion of vehicles in specific lanes. In addition to the static policy, dynamic control system uses detectors/sensors to detect cars on roads or intersections and changes the phase with respect to the information acquired from detectors to adjust the traffic flow [4]. It is much more efficient than the static policy with fixed time system, however, it still lacks substantial adaptability to changes in traffic. While certain general statements about the levels of traffic at various times tend to hold true, minor differences are always present from moment to moment. It is affected by many variables, such as general stochastic nature of human decision making, events in the area, accidents, or even the weather making the traffic optimization problem highly complex.

In order to optimize traffic flow in consideration of such uncertainties, there has been many cases of individuals trying to make intelligent traffic light control system by using artificial intelligence methods [5]. Among them, reinforcement learning (RL) has been attracting attention to solve traffic light optimization problems. RL has abilities to 1) adapt to different situations that are caused by weather conditions, car accidents, and special events 2) learn to optimize traffic flow that is difficult to give a correct answer without supervision by human experts 3) model environment on its own, since an agent learns every variable of the environment through learning procedure. Although these advantages have been pushing RL forward for traffic light control problems and research works so far have successfully eased traffic congestion in simulation environments, these methodologies have not been utilized with actual traffic lights. One of the problems to practically use RL for traffic light optimization is that these methods require huge amount of data for training to make effective implementation in reality which is difficult [5].

This project implements RL for optimization of traffic light control at 4-way intersection where traffic is generally the heaviest, but more practical than other past works in terms of efficiency, allowing for these methods to be employed to learn off of real world data. Specifically, we devise a time-efficient optimization strategy by combining transfer learning and RL. The system first trains to imitate the fixed sequence traffic policy and then transfers the learned features to our RL target models. These models then use evolutionary algorithms or actor critic RL to further improve the model. Data for training our system is collected by running simulations using Simulation of Urban Mobility (SUMO) software [6]. The input for our system is the current state of the intersection, which consists of the positions and speeds of cars, and traffic light phase. Using this input, our system outputs the next phase the traffic system should transition to.

II. Background

A. Neural Networks

Neural networks are representations of functional mappings between inputs and outputs that have gained increasing popularity in the last decade. They propagate inputs through some number of computational layers, with the output of each layer fed into the input of the next before eventually generating an output. A key feature of these networks is their ability to represent a function of arbitrary complexity. The universal approximator theorem states that a neural network with at least two layers can be constructed to approximate *any* function within a given finite error [7]. This ability to represent any function removes the need to make assumptions about the form of the mapping between the inputs and outputs that is desired to be learned. Given that the mapping between the traffic state and best traffic control choices is highly complex, neural networks pose a convenient tool to use for optimal traffic control. Furthermore, optimization algorithms have been developed (such as Adam[8]) that enable the parameters of neural networks to be tuned to best represent the function we are attempting to discover. We can take advantage of these algorithms as we teach our neural network controller.

B. Imitation Learning

Imitation learning is a form of supervised learning where the goal is to develop a model to imitate the behavior of some entity. People have attempted to use imitation learning in automated driving since the 1980s, where human driver responses are the imitation goals [9]. For our purposes, the response of a sequential traffic light sequence is what we hope to imitate. This provides us with a baseline performance for comparison and the starting point for transfer learning.

C. Neuroevolutionary Learning

Neuroevolution is the combination of neural networks with evolutionary algorithms. Evolutionary algorithms are a form of population based algorithms for optimization. Neuroevolution then uses neural networks as the members of the population for a given evolutionary algorithm. There is not one defined flavor of evolutionary algorithms[10], but they consist of two fundamental components: mutation and selection. In mutation, additional members are added to the population by making random modifications to existing members. In neuroevolution, this consists of adding some noise to some fraction of the weights of neural network. Once mutation is done, a fitness value is retrieved for each network by an evaluation of its performance. Selection of which networks continue in the algorithm is then made, with higher fitness values corresponding to improved chances of proceeding to the next generation. This continues in a loop to gradually explore the neural network weight space in an attempt to find an optimal network.

D. Advantage Actor Critic

Actor critic methods are a form of reinforcement learning that use two models to inform the learning process [11]. The actor is a model that maps inputs to actions, while the critic is a model that maps inputs to values. The learning process then consists of updating the actor model using values from the critic as weighting factors. In Advantage Actor Critic, a value function (learned from discounted rewards) is used a baseline to influence the updating of the actor weights via equation 1.

$$\theta := \theta + \alpha \sum_{t=1}^{T} (r_t + \gamma V(s') - V^{\pi}(s)) \nabla_{\theta} log \pi(a_t | s_t; \theta) \tag{1}$$

where θ is the weights of the actor network, V is the value function, s is the current state, s' is the next state, a is an action, π is the actor policy, α is the learning rate, and γ is the discount factor.

This and other actor critic methods can take advantage of intraepisode results to modify their policies, allowing for improved learning speed over evolutionary algorithms, with a decrease in the stability of the solutions provided.

E. Transfer Learning

Transfer learning is a subsection of machine learning where information learned for one task is applied to another task. If the two tasks are similar enough, then the speed of learning the second task can be increased over starting from scratch. Transfer learning has been utilized in domains such as image classification[12], but remains unutilized in traffic optimization literature.

A simple approach to using transfer learning is to use some or all of the trained weights from a neural network at the start of training of a network for a similar task. This is how we utilize this transfer learning.

III. Related Work

Intelligent traffic light control methods have been studied for improving conventional traffic systems to be more adaptable to dynamically-changing traffic flows in the real world. Chiu et al. [13] presented an optimization method for traffic signal timing using fuzzy logic. Schutter et al. [14] proposed a model that describes the evolution of the queue length at an intersection for solving traffic light switching problems. These studies, however, model road traffic by limited information so that these approaches cannot be applied in complex, realistic traffic control problems.

Learning-based traffic light control has been actively studied to optimize dynamic, uncertain traffic situations in the real world. Liu [15] summarized intelligent traffic signal control methods until 2006, and RL is introduced as one of the effective optimization technologies. During this period, however, RL methods were still under development, such as using a linear function to estimate the Q values. In that period, a small-sized state space was mainly applied for RL. Abdulhai et al. [16] presented an intelligent transportation system by Q-learning that represents state space based on the number of waiting vehicles. Balaji et al. proposed multi-agent traffic signal control by RL that builds its state space using vehicle occupancy and statistics obtained from historical traffic data. The difficulty of traffic light control by RL in these methods is that the dynamic, complex urban traffic systems cannot be represented by the limited information.

Recent technological advancements in the field of artificial intelligence have been pushing the use of both deep learning and reinforcement learning as deep-reinforcement learning to estimate Q-values for policy implementation. Several research works apply deep-reinforcement learning to control traffic lights in uncertain, dynamic environments. Li et al. [17] used deep reinforcement learning to find appropriate signal timing policies. Pol et al. [18] presented a traffic light control approach by deep Q-learning to optimize traffic flow in the cross-shape intersection. Pol represents the state of traffic as an image matrix, and the reward is designed by combining vehicle and traffic light information such as waiting time and whether a light changes or not. Liang et al. [19] applied a deep-reinforcement learning model consisting of several components such as a dueling network and double Q-learning network to improve optimization performance. Gao et al. [20] proposed an algorithm that automatically extracts useful features for deep reinforcement learning to learn an optimal policy, instead of using human-created features. The main motivation of this approach is to gain useful information from raw data that is sometimes ignored when human experts arbitrarily select information. Above all the research works based on deep reinforcement learning show better performance in terms of optimizing traffic flow with complexity.

However, there are still problems in accomplishing optimized traffic light control using reinforcement learning. The first problem is the design of a reward that is suitable for optimizing traffic flow. The objective of the traffic light to optimize traffic flow is to minimize total travel times of vehicles on roads. However, the travel time of vehicles cannot be obtained until they pass the roads; hence, it leads to the problem of delayed rewards. Instead of this as a reward, the change of the cumulative waiting time and combination of cars' rewards such as sudden braking and deceleration have been used as rewards [18] [19]. Appropriate design of rewards is one essential aspect of the traffic light control learning that is not fully solved.

The next problem is that current RL-based approaches do not consider efficiency of learning processes. While these methods perform better results to solve traffic congestion, the learning time is too long for optimization in the real world. During trial-and-error process of RL, vehicles in the real world will get caught in traffic jams in the early stages of RL traffic light control. Hence, traffic light control policies are required to quickly adapt the real world situations and even avoid these initial problems. In order to solve this problem, time-efficient optimization method by RL is required such as learning from limited data samples, efficient exploration methods, and implementing it with transfer learning [5]. The approach in this project focuses on the fact: traditional traffic light systems are able to manage most traffic flows decently, but fails to provide good performance for specific scenarios. Hence, the traditional traffic light policy is utilized as a reference for the system to learn.

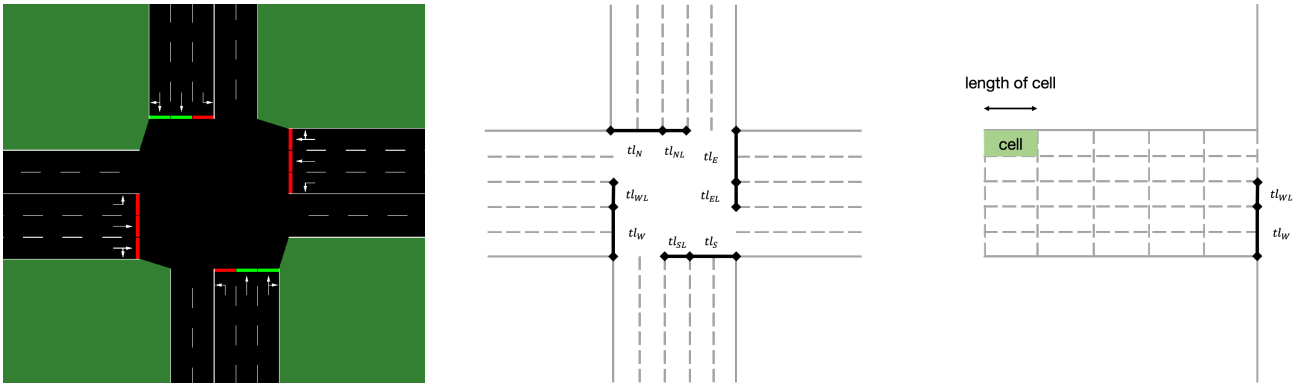
IV. Methodology

A. Simulation Environment

The simulated environment is a four-way intersection using Simulation of Urban MObility (SUMO) as shown in Figure 1 (a). It consists of a four-way intersection with three incoming lanes on incoming road. The three lanes for each road can go left, straight, or straight and right. The length of each lane from vehicles' origin to the intersection's stop line is 1000 m, while policies implemented can detect cars within 50 m of the intersection. Traffic lights are represented by the color on the stop line of every incoming lanes. The environment has eight traffic lights that controls one or more adjacent lanes. The set of traffic lights are represented as,

$$TL = (tl_W, tl_{WL}, tl_S, tl_{SL}, tl_E, tl_{EL}, tl_N, tl_{NL}), \tag{2}$$

where tl expresses a traffic light for one or two lanes, subscripts express the position of these traffic lights as shown in Figure 1 (b). The states of all of these traffic lights are composed of *green*, *yellow*, and *red*. A *Green* light denotes that vehicles may pass the intersection if no other vehicles with higher priority will intersect their path, *yellow* lights denotes that vehicles start to decelerate if appropriate, and *red* light denotes that vehicles must stop.



(a) (b) (c)

Fig. 1. (a) Simulation environment using SUMO software. Arrows on each lane expresses the way vehicles can pass through such as turn left, right, and go straight. (b) Positions of traffic lights in the environment. tl expresses a traffic light for one or two lanes and subscripts express the position of these traffic lights. (c) A method to express vehicles' positions as a vector using cells' information.

Traffic flow on each lane is generated by setting the probability of vehicle spawn in each lane to have control to represent specific traffic conditions. In the simulation environment, the specifications of each vehicle are set as acceleration ability, deceleration ability, and maximum speed being $1.0m/s^2$, $4.5m/s^2$, and $25m/s$ with length of each vehicle being $5m$. From all vehicles on incoming lanes, vehicles' positions, speeds, and duration of the stopped cars are retrieved to represent the state and reward as mentioned in the following sections.

1) State Representation: The state for the agent is a representation of the intersection environment for a certain time step. It is designed to sufficiently provide information for the agent to learn the optimization policy for the traffic flow as follows,

$$state = (position, speed, phase) \tag{3}$$

where $position$ denotes the position of each vehicle in each lane, $speed$ denotes the speed of each vehicle, and $phase$ denotes the phase and current length of time spent in that phase. The whole intersection is divided into uniform cells as shown in Figure 1 (c). $Position$ is expressed as a binary value to show whether there is a vehicle in the cell or not. $Speed$ is represented as a value of m/s in cells where there are vehicles. On the other hand, $phase$ expresses which phase the traffic light is from the defined ten actions as mentioned in the following section. Hence, the state is expressed as a vector format as eq. 3.

2) Action Set: The action set is designed to express which color the traffic lights to turn on to control traffic flow. Agent of this environment is the traffic light system on the intersection so that the new action set is chosen, the traffic lights change colors. Ten action sets are generated as 1) South and North straight green, 2) West and East straight green, 3) South and North left turn green, 4) West and East straight green, 5) West straight and left turn green, 6) East straight and left turn green, 7) South straight and left turn green, 8) North straight and left turn green, 9) East and West straight and yield left turn green, and 10) North and South straight and yield turn green, as shown in Figure 2. The traffic light colors change from red to green directly, but has to go through yellow when changing from green to red color as real world. Hence, the yellow light phase is performed when the chosen action is different from previous action.

3) Reward: The reward is the feedback from the environment after the agent chose and perform the action. Appropriate reward design is important for the agent to learn best action policy. In the target environment, the purpose of learning process is to increase the efficiency of the intersection with reducing the waiting time of vehicles. Hence, the reward is defined as the change of the cumulative waiting time between two neighboring time steps. The total waiting time of step t is expressed as

$$W_t = \sum_{i_t=1}^{N_t} w_{i_t,t}, \tag{4}$$

where $w_{i_t,t}$ denotes a waiting time of one vehicle at time step t, N_t denotes the total number of vehicles at t. The reward is expressed by combining eq. 4 at t and $t-1$ as

$$r_t = W_{t-1} - W_t. \tag{5}$$

The reward in eq. 5 is the increment in cumulative waiting time between before and after taking action. If the W_{t-1} is better than W_t, the reward becomes positive value. On the other hand, if the W_{t-1} is worse than W_t, the reward becomes negative. Only the negative reward is accumulated during the episode as a penalty to the bad action of the agent. Hence, the agent learns a good policy by minimizing the penalty through learning process.

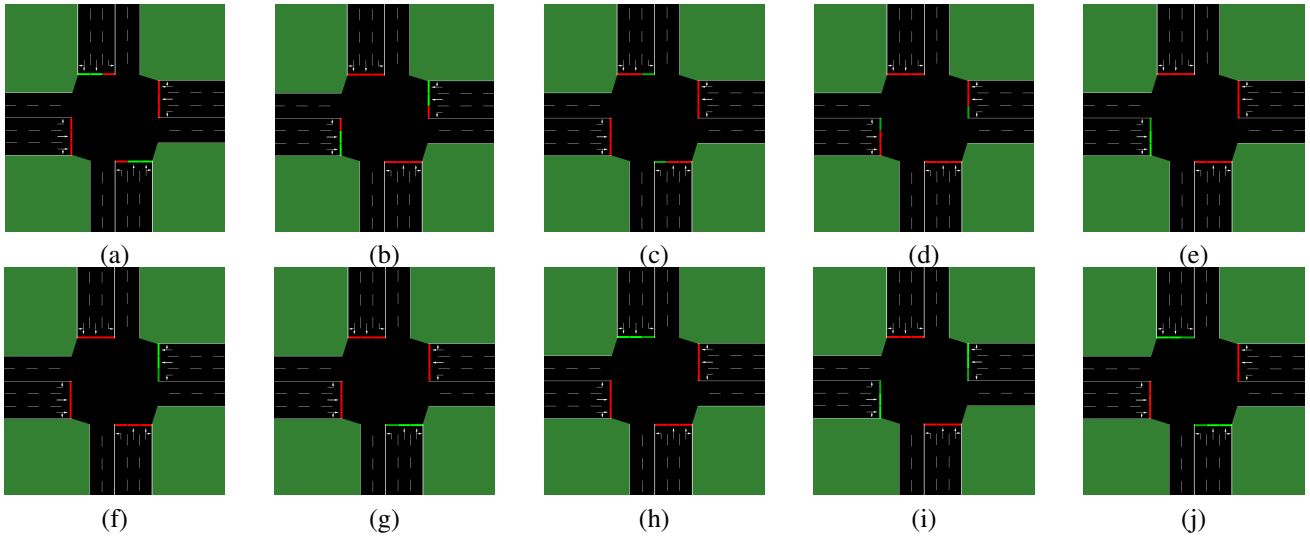


Fig. 2. (a) South and North straight green, (b) West and East straight green, (c) South and North left turn green, (d) West and East straight green, (e) West straight and left turn green, (f) East straight and left turn green, (g) South straight and left turn green, (h) North straight and left turn green, (i) East and West straight and yield left turn green, and (j) North and South straight and yield turn green

B. Learning Mechanisms

1) Imitation Learning: In order to accomplish imitation of the fixed time traffic sequence, we gather data for a large number of episodes where the sequential policy was used. This data of $(state, action)$ pairs are then used to train a neural network policy with supervised learning. Currently, only a single traffic distribution has been trained on, but we plan on expanding this training to include different traffic distributions to ensure consistency in the trained network.

2) Neuroevolutionary Learning: For evolutionary learning, k randomly weighted neural networks make up the starting population. In the mutation phase, each neural network has f of their weights adjusted according to the normal distribution $\mathcal{N}(0, \sigma^2)$. The networks are then used as the policy for an entire episode of simulation, with the cumulative reward for each episode being that network's fitness. The networks are then randomly paired in a binary tournament, where the network in each pair that has a higher fitness has a likelihood of $(1 - \epsilon)$ of being selected and the lower fitness network has a likelihood of ϵ of being chosen. The selected networks continue in the evolution process while the others are removed. This process continues for the number of generations specified for a given experiment.

3) Advantage Actor Critic: For our implementation of Advantage Actor Critic, we use batch updates to our actor and critic. For a given batch size, the $(state, action, reward)$ tuples are stored until the batch is full, then the critics value estimate is utilize in the updating of the weights for both the actor and critic network. This enables the actor to improve in the middle of an episode.

4) Transfer Learning: We used the network trained with imitation learning as the starting point for both neuroevolutionary and the actor in advantage actor critic learning methods. This is with the goal of enabling better lower bound performance during learning and improving the speed of learning better solutions.

V. Results

A. Experimental Setup

Experiments are run for 1800 simulated seconds (30 minutes) in a SUMO simulation environment, with an expected 1500 cars being generated with a random uniform distribution in each incoming path. The environment queries a policy every 5 seconds of green light for the next phase to change to. This policy is either a fixed sequential policy that chooses to run each light phase for 30 seconds, or some neural network that makes its decision based on its mappings from the state to its output. If the next phase is the same, no lights change, but if the next phase is different, 4 seconds of yellow lights occur before switching the new phase to green. The parameters for specific algorithms are shown in Table I.

TABLE I
List of hyper-parameters of learning algorithms

Type of algorithm	Imitation Learning	Evolutionary Learning	Advantage Actor Critic
Hyper-parameters	Number of episodes for training data: 500	k: 5	T: 10
		f: 0.1	γ: 0.9
	Number of training epochs: 10	σ: 0.1	
		ϵ: 0.1	α: 0.001

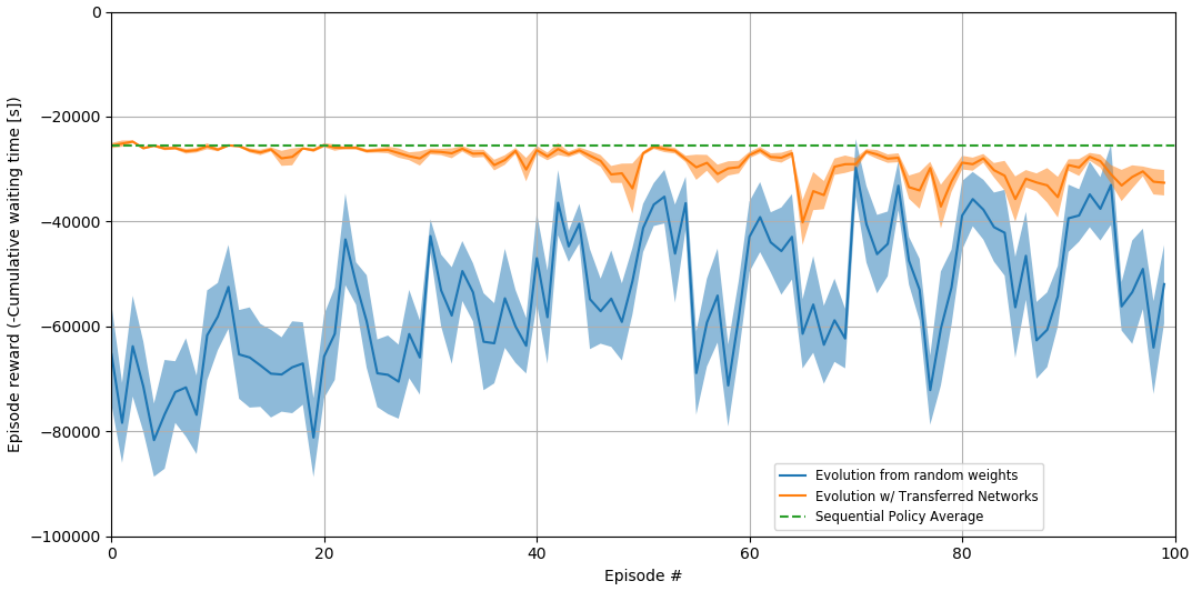


Fig. 3. Experimental results for evolutionary approach from random weights and transferred networks. The average total reward $\pm$ the standard error over 11 trials is what is shown, alongside the average over 100 episodes for the constant sequential policy for comparison.

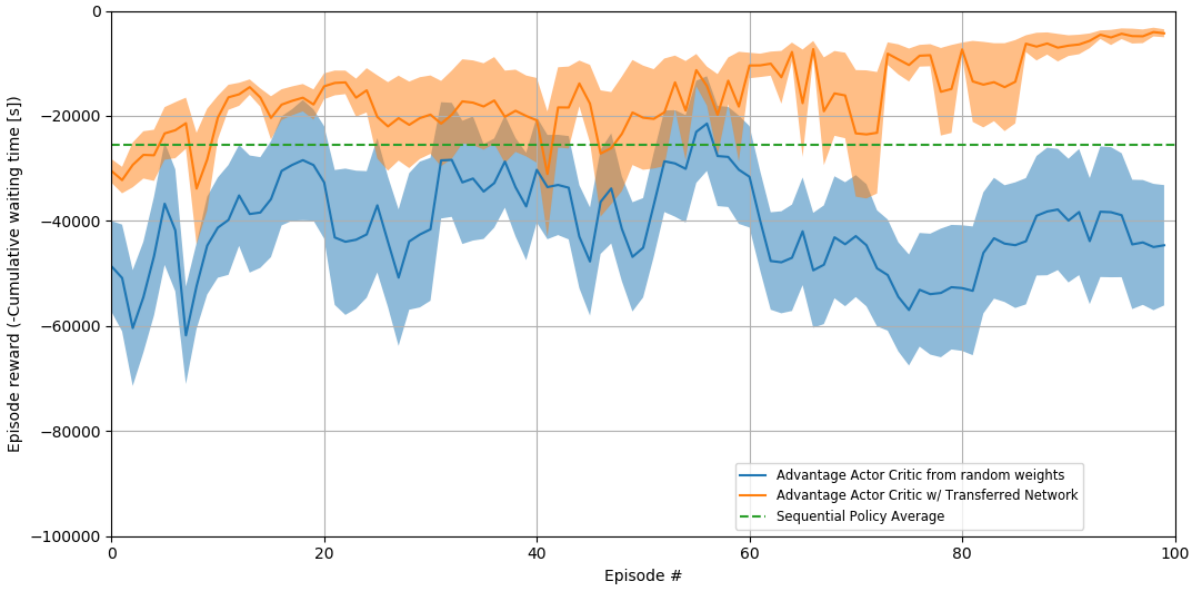


Fig. 4. Experimental results for advantage actor critic approach from random weights and with a transferred actor. The average total reward $\pm$ the standard error over 11 trials is what is shown, alongside the average over 100 episodes for the constant sequential policy for comparison.

B. Results

Experimental results are shown in figure 3 and 4. These plots show the cumulative reward received at the end of a given training episode over 100 such training episodes, with the standard error of the mean being used for the upper and lower bounds of each point. The green dotted line expresses the average cumulative reward of the sequential policy. This line implies the performance of the traditional traffic lights policy so that learned policies can be assessed as whether they can improve their performance than typical traffic lights. Note that the evolutionary plots are scaled where a ”training episode” is any episode run by a network in the population. Thus, a generation is marked by every 10 episodes. Higher rewards represent

better performance, and the average over 100 episodes of the sequential policy is included as a baseline reference.

From examining figure 3, we can see that the beginning behavior is what is expected for both the randomly weighted and transferred networks. The randomly weighted networks perform rather poorly at the beginning, managing to almost have some individuals achieve sequential performance by the end of training. The trend of improvement would continue if allowed more time, but that is not relevant to the results we are interested in. The transferred networks began with performance nearly identical to the hard-coded sequential policy. This is as expected because the networks were trained to imitate that policy. But as training went on, the results became worse instead of improving. This is because the starting networks provided the algorithm with no variation in its population, and all of its members were in a likely local maximum, making improvement difficult. The noise added to the network weights then produced a muddied up sequential policy rather than any novel control strategies.

Looking at figure 4, we can see how the advantage actor-critic training fared with a randomly initialized or transferred actor. We can see that the transferred actor improves more consistently over the training episodes than its randomly initialized counterpart. The transferred networks also manage to consistently converge to a good policy while the randomly initialized network on average ends roughly where it started and with a large amount of variation. With that said, there was still notable amounts of variation in the performance from the transferred network. We believe that our training may have not adequately filled the input space or otherwise overfit the training data. This could potentially cause a positive feedback loop where the policy is stuck causing the state it failed to interpret properly over the entire episode.

VI. Discussion and Conclusion

A. Discussion

Our results show that transferring a simplistic traffic control strategy to be the starting point for another learning algorithm can be helpful or detrimental. Evolutionary algorithms rely heavily on variance in their population, and by removing that by beginning with a single policy, learning is much more difficult. If that policy is a local maximum, the odds of successful exploration are even lower. On the other hand, beginning with a transferred network for actor-critic methods makes the convergence of the algorithm less brittle by providing it with some usable starting information. This is not to say that tuning learning parameters cannot improve performance from this point, but it helps make this tuning a less necessary to converge to good performance. In both cases, transferring provides a significant boost to the starting performance during learning. This starting performance is essential if these learning algorithms are desired to be performed using real life systems.

B. Conclusion

We present a traffic light control learning strategy that transfers an imitation of a basic sequential policy to other learning methods. This allows for the starting point of learning to be improved and require less training. We performed experiments to validate this for a 4-way intersection under moderate, uniform traffic, finding that the effectiveness of transferring policy is dependent on the algorithm used to learn post-transfer. Evolutionary algorithms fail for this purpose while actor-critic does not. Overall, the contribution of this paper is:

- Utilizing an existing static traffic control policy to improve initial performance, consistency of learning and convergence of final traffic control policy during reinforcement learning

C. Future Work

There are multiple directions to continue this line of research. First is performing experiments similar to what we performed in this paper, but with various different traffic conditions. We only examined moderate traffic flow that was uniform from all directions. Lighter and heavier traffic flow, as well as non-uniformly distributed traffic flow would be worth examining to fully explore possibilities with minimal setup changes. Second would be to explore other traffic policies that are less simplistic than the sequential timed policy used in this paper for imitation. Inclusion of induction loop logic or even more complex algorithms for imitation can provide the starting networks with more information and prevent the dead sections that arise in the networks from ignoring everything but the time in the current phase. A third direction is to explore transfer learning into other reinforcement learning frameworks. We explicitly explored two specific types of learning post-imitation, but there are many more forms out there. Lastly, these techniques could be applied to show reinforcement learning performing on a real world system instead of in simulation.

References

[1] History.com Editors, (2009, November 13) “First electric traffic signal installed,” *A & E Television Networks*, Retrivied from https://www.history.com/this-day-in-history/first-electric-traffic-signal-installed.

[2] Nelson, M. K. (2018, May 1). A Brief History of the Stoplight. Retrieved from https://www.smithsonianmag.com/innovation/brief-history-stoplight-180968734/.

[3] The Economist. 2014. The cost of traffic jams. https://www.economist.com/blogs/economist-explains/2014/11/economist-explains-1, November, 2014.

[4] P. B. Hunt, D. I. Robertson, R. D. Bretherton, and M. C. Royle, "The SCOOT on-line traffic signal optimization technique," *Traffic Engineering & Control*, Vol. **23**, No. 4, pp. 190–192, 1982.
[5] H. Wei, G. Zheng, V. Gayah, and Z. Li, "A survey on traffic signal control methods," *arXiv preprint arXiv: 1904.08117v2*, 2019.
[6] K. Krajzewicz, J. Erdmann, M. Behrisch, and L. Bieker, "Recent development and applications of sumo-simulation of urban mobility," *International Journal on Advances in Systems and Measurements*, Vol. **5**, No. 3&4, pp. 128–138, 2012.
[7] K. Hornik, M. Stinchcombe, and H. White "Multilayer feedforward networks are universal approximators," *Neural Networks*, 1989.
[8] Yoshua Bengio and Yann LeCun, "Adam: A Method for Stochastic Optimization," 3rd International Conference on Learning Representations, San Diego, CA, USA, May 7-9, 2015.
[9] Dean A Pomerleau. Alvinn, "An autonomous land vehicle in a neural network. In Advances in Neural Information Processing Systems," pages 305–313, 1989.
[10] Zelinka Ivan, "A survey on evolutionary algorithms dynamicsand its complexity–Mutual relations, past, present and future," Swarm and Evolutionary Computation 25, 2-14, 2015.
[11] Asadi et. al., "Mean Actor Critic," arXiv:1709.00503v1 [stat.ML] 1 Sep 2017.
[12] Zhu et. al., "Heterogeneous Transfer Learning for Image Classification," Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence, 2011.
[13] S. Chiu and S. Chand, "Adaptive traffic signal control using fuzzy logic," *Proceedings. The First IEEE Regional Conference on Aerospace Control Systems*, Westlake Village, CA, USA, 1993, pp. 122–126.
[14] B. De Schutter and B. De Moor, "Optimal traffic light control for a single intersection," *European Journal of Control*, Vol. **4**, No. 3, pp. 260–276, 1998.
[15] Z. Liu, "A survey of intelligence methods in urban traffic signal control," *IJCSNS International Journal of Computer Science and Network Security*, Vol. **7**, No. 7, pp. 105–112, 2007.
[16] B. Abdulhai, R. Pringle, and G. J. Karakoulas, "Reinforcement learning for true adaptive traffic signal control," *Journal of Transportation Engineering*, Vol. **129**, No. 3, pp. 278–285, 2003.
[17] L. Li, Y. Lv., and F. Y.Wang, "Traffic signal timing via deep reinforcement learning," *IEEE/CAA Journal of Automatica Sinica*, Vol. **3**, No. 3, pp.247–454, 2016.
[18] E. van del Pol and F. A. Oliehoek, "Coordinated deep reinforcement learners for traffic light control," *NIP's Workshop on Learning, Interfaces and Control of Multi-Agent Systems*, 2016.
[19] X. Liang, X. Du, G. Wang, Z. Han, "A deep reinforcement learning network for traffic light cycle control," *IEEE Transactions on Vehicular Technology*, Vol. **68**, No. 2. pp. 1243–1253, 2019.
[20] J. Gao, Y. Shen, J. Liu, M. Ito, and N. Shiratori, "Adaptive traffic signal control: deep reinforcement learning algorithm with experience replay and target network," *arXiv preprint arXiv: 1705.02755*, 2017.