Computer Standards & Interfaces 87 (2024) 103771

Contents lists available at ScienceDirect

Computer Standards & Interfaces

journal homepage: www.elsevier.com/locate/csi

Intelligent traffic light systems using edge flow predictions

Adam Rizvi Thahir [a], Mustafa Coşkun [b,*], Sultan Kübra Kılıç [a], Vehbi Cagri Gungor [a]

[a] Electrical & Computer Engineering, Abdullah Gul University, Kayseri, Turkey
[b] Artificial Intelligence and Data Engineering, Ankara University, Ankara, Turkey

ARTICLE INFO

Keywords:
Artificial intelligence
Reinforcement learning
Traffic flow
Congestion

ABSTRACT

In this paper, we propose a novel graph-based semi-supervised learning approach for traffic light management in multiple intersections. Specifically, the basic premise behind our paper is that if we know some of the occupied roads and predict which roads will be congested, we can dynamically change traffic lights at the intersections that are connected to the roads anticipated to be congested. Comparative performance evaluations show that the proposed approach can produce comparable average vehicle waiting time and reduce the training/learning time of learning adequate traffic light configurations for all intersections within a few seconds, while a deep learning-based approach can be trained in a few days for learning similar light configurations.

1. Introduction

With the advent of mega-cities, the traffic congestion problem has been ever-increasing and introducing additional problems, such as time consumption, air pollution, and economic burdens. For example, an average driver in the United States had lost 99 h in traffic congestion in 2019 with an estimated cost of $ 1,377 [1]. Moreover, the traffic congestion associated with time consumption and cost can be more severe in large cities, such as in Boston, up to 149 h with an estimated cost of $ 2,205 [1]. Thus, releasing traffic congestion by utilizing "adaptable" traffic management systems has been one of the central research interests.

In today's traffic management systems, most of the cities operate their traffic lights in a predefined fixed time fashion [2,3] or design manual hand-crafted traffic rules by observing traffic in real-time [4]. However, predefined rule-based traffic management systems are vulnerable to changes that are inevitable due to the enlargement of the cities and the increase in the number of vehicles on the roads. As a result, these hand-crafted rule-based systems are often deemed to be limited [5].

Inspired by the fascinating developments in Artificial Intelligence (AI), recent research attempts based on AI approaches aim at solving traffic management problems with the hope of learning adaptive traffic light configurations for intersections. To this end, the earlier studies have proposed to use traditional AI approaches, such as reinforcement learning (RL) algorithms [6–8]. In this context, the traffic management systems (TMSs) are stated as an RL problem by treating each intersection as an intelligent agent. The overall flow of the usage of the RL in the context of TMS is depicted in Fig. 1. More specifically, A TMS consists of three main parts:

- Environment: composed of traffic light phases and traffic conditions, including congestion and traffic outward flow.
- State: a feature representation of the environment, usually a grid representation of an intersection to a certain extent of the intersection.
- Agent: creates a model which is able to make a decision about the light configuration based on the input that is sent from the environment.

Concretely, following the decision-making process of the agent, the environment returns a reward value back to the state. Then, this reward is used as an additional input to the agent, the model updates its parameters to guide the agent and to make more optimal decisions. However, RL-based approaches can suffer from the large state space problem [9]. Thus, more recent AI-based approaches utilize function approximation of RL through deep neural network learning [5,9–11] to circumvent the large state space problem.

Despite the effectiveness of the existing deep learning-based TMS [5,9–11], when the number of intersections increases, the training time of the neural networks to find the optimum parameters for traffic light adjustment can be computationally costly. To alleviate this problem, in this paper, we propose to use graph-based semi-supervised learning for edge label prediction [12]. More specifically, we treat the TMS problem as a graph problem where intersections are represented by nodes and

* Corresponding author.
E-mail addresses: adam.thahir@agu.edu.tr (A.R. Thahir), coskunmustafa@ankara.edu.tr, mustafa.coskun@agu.edu.tr (M. Coşkun), sultankubra.kilic@agu.edu.tr (S.K. Kılıç), cagri.gungor@agu.edu.tr (V.C. Gungor).

https://doi.org/10.1016/j.csi.2023.103771
Received 26 February 2023; Received in revised form 5 June 2023; Accepted 29 June 2023
Available online 4 July 2023
0920-5489/© 2023 Elsevier B.V. All rights reserved.

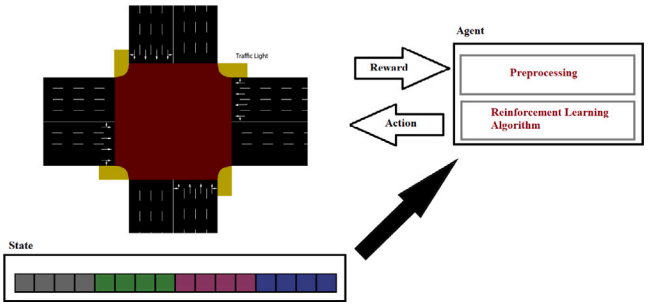

Fig. 1. Visualization on how a generic Reinforcement Learning algorithm would work on a traffic light environment.

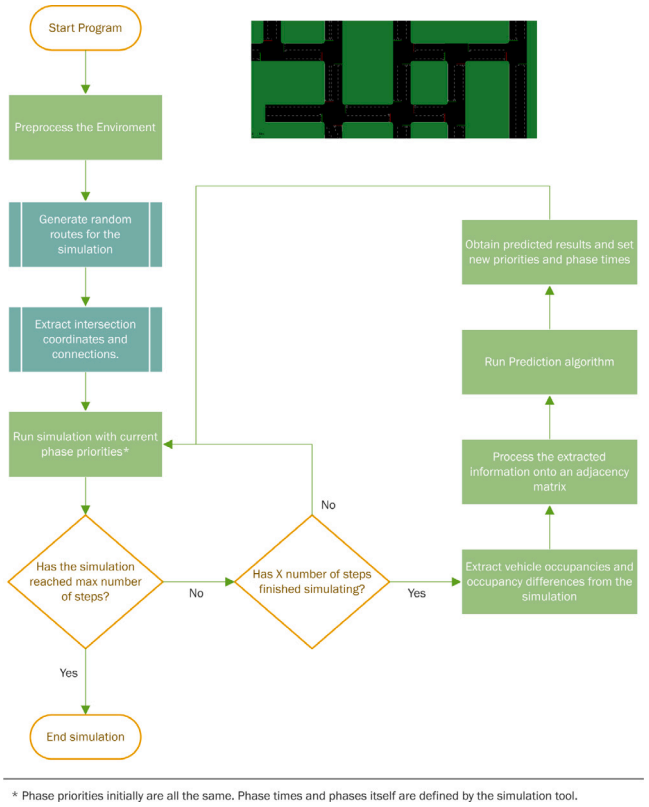

Fig. 2. Flowchart representation breaking down the steps on how the proposed algorithm is set to work.

the roads connecting them are represented as edges in the graph. We then translate the TMS problem to a flow prediction problem on a graph. Ideally, if we know that some of the roads are congested by the load of the traffic, we aim at predicting which roads would be most affected by this flow of congestion. To this end, we use an edge label propagation algorithm, where an edge label of 1 indicates that the road is congested; 0 otherwise. [12].

To assess the performance of our proposed approach against state-of-of-the-art deep learning approaches as well as heuristic-based approaches, we conduct our experiments on Simulation of Urban MObility (SUMO) platform [13] by selecting average vehicle waiting time as an evaluation metric. Specifically, we compare our proposed approach against two deep learning-based algorithms A2C [14] and PPO [15] as well as two heuristic-based approaches, based on vehicle occupancy [5] and scoring. Comparative performance evaluations on various traffic intersection configurations show that our approach can produce comparable average vehicle waiting time and drastically reduce the training/learning time of learning adequate traffic light configurations for all intersections within a few seconds, while deep learning-based approach can be trained in a few days for learning similar light configurations. Fig. 2 visualizes the general workflow of the proposed approach. Our contributions can be summarized as follows:

- We develop a system that is able to predict traffic flow between multiple intersections using Edge-based Semi-Supervised Active Learning.
- Our approach learns traffic light configuration in a few seconds while deep learning-based approaches can learn similar light configurations in a few days.
- By assessing the average waiting times for vehicles in a traffic environment, we observe that, in a few seconds, our method can yield better than that of heuristic approaches and comparable results to that of deep learning approaches, which last a few days.

The rest of the paper is organized as follows. Section 2 discusses related work and other approaches. Section 3 formally defines the problem. Section 4 explains the methodology. Section 5 explains the results obtained from the simulations conducted. Finally, Section 6 concludes the study.

2. Related work

We can broadly divide traffic management system development into three categories: (i) data collection, (ii) data processing/cleaning, and (iii) optimization. The data collection and processing are two consensus processes in the traffic management system, as the gathered information in these two steps is used to understand vehicle queues and occupancy at the intersections. Studies [19–23] focus on areas related to communication with intelligent traffic management systems at different network layers as well as hardware levels. The study [19] employs a system with a number of layers within the system in order to ensure data reliability as well as security of the data collection and processing phases. The works of [20,21] propose a system where hardware devices are used in order to store and exchange information with a given intelligent traffic light system. In the literature, there are also data collection techniques, including image processing [24,25] and inductive loops [26,27], to name a few.

The final phase, optimization of the traffic management system, aims at solving the traffic congestion problems by taking the number of vehicles, and queue sizes into account. Thus, the research developed for optimizing traffic management systems, which is also the interest of this paper, can be viewed in two broad categories: (ii) Conventional Traffic Light Control Systems and (iii) Reinforcement Learning (Deep Learning) based Traffic Light Control Systems.

2.1. Conventional traffic control systems

Earlier approaches for traffic light control systems aim to optimize waiting time in traffic intersections [2,16]. These approaches are based on the observation of current traffic flow and they set manual rules for traffic lights accordingly and then they use these sets of rules for intersections. Similar systems have been conducted in the studies of [2,3,16]. Another conventional approach is to use real-time traffic data when processing the optimal times for a given traffic phase [4,28]. Although these types of traffic management systems can offer certain merit in the management of traffic lights, the calculated optimal times could become redundant due to out-of-date traffic data, or would not be able to reach the optimal times due to not having any insights on future traffic data.

Table 1
Comparison table.

	Simulation	Predict flows	Real time data	Historical data	MultiIntersection
IntelliLight[11]	✓	X	✓(online)	✓(offline)	X
Francois Dion et al [16]	X	X	X	✓	X
Alan J Miller[2]	X	X	X	✓	X
Brian L, Smith [17]	X	✓	X	✓	X
FRAP [18]	✓	X	✓	X	✓
CoLight [10]	✓	X	✓	X	✓
Our Approach	✓	✓	✓	✓	✓

2.2. Reinforcement learning for traffic light control

As the traffic management system problem is a time-dependent dynamic problem, with the development of AI-based algorithms, recently Reinforcement Learning/Deep Learning approaches [5], have been employed to dynamically adjust the traffic lights by learning traffic behaviors.

Previous studies relate traffic data with the state space of the environment. [6,7,29–31] use the number of vehicles queued at a given road as their state space. [32,33] use traffic flow obtained by sensors exiting a given intersection. Action spaces are defined based on signal phases, which can include all available signal phases as seen in [33,34] or only the relevant green signal phase, as seen in [32] and [30]. Reward functions of the environments are typically related with how to direct change could be observed from actions taken, [33,34] define their reward function based on the change in delay for the vehicles in the environment, similarly, [30,32] define their reward function based on the change in the number of queued vehicles at given roads in the intersection.

The agents in a reinforcement learning solution act on the environment based on the actions defined in the action space of the environment, the observation of the action taken can be viewed from the state space, and the reward for each action taken is calculated from the reward function. The Deep Q learning and policy gradient approach was studied in [5], which proposes using the reward function as traffic flow and traffic delay of vehicles in the network. Moreover, other deep learning-based approaches have been studied in the Stable Baselines project [35], using the Asynchronous method for Deep Reinforcement Learning (A2C) [14] and Proximal Policy optimization Algorithm [15].

Comparing the traditional methods [7,32] in the context of traffic management systems, deep learning-based approaches [5,10,11,35] have been repeatedly shown to be more effective since these methods can learn dynamic nature of the traffic within trail-and-error based approaches by updating the neural network parameters. However, in these deep learning algorithms, training the neural network parameters is computationally expensive as the number of intersections increases.

In this paper, we propose an alternative and efficient algorithm by stating the TMS as a flow prediction problem. Our approach renders predicting the outward flow of vehicles from a given intersection and enables the TMS system to identify vehicle congestion that would possibly occur at a following intersection. Thus, our method prepares the intersection that is anticipated to be congested. More specifically, our approach in the multi-intersection setting can provide information on where the traffic flow takes place, pinpointing the intersections that are going to be congested in the order of seconds (efficiently) based on semi-supervised active learning on edges [12]. Here, the main difference of our proposed approach over other deep learning or traditional methods is that the flow of traffic can be anticipated beforehand. Hence, the intersections, that are predicted to be congested, can adjust their traffic lights accordingly. More importantly, since we treat each intersection as a node and a large city can have traffic light intersections in the range of thousands, our graph-based approach can efficiently process these nodes (intersections). Thus, our approach can scale to a city-level traffic management system.

To assess the performance of our proposed approach against state-of-the-art deep learning approaches, such as Asynchronous method for Deep Reinforcement Learning (A2C) [14] and Proximal Policy optimization Algorithm [15], as well as two heuristic-based approaches [36], we conduct our experiments on Simulation of Urban Mobility (SUMO) [13] platform by selecting average vehicle waiting time as an evaluation metric. The first heuristic-based approach simply decides and prioritizes the next traffic light phase based on the current vehicle occupancy at a given intersection. The other heuristic-based approach is based on a local scoring mechanism [36] and changes the traffic phase and priority based on local scores. For a better visualization, Table 1 shows a comparison table showing differences between our approach and the existing recent studies focusing on traffic light management systems.

3. Methods

In this section, we first define the "intelligent" traffic management system problem. We then give a brief background on state-of-the-art approaches to solve this problem, i.e., heuristic and deep learning-based approaches, respectively. We finally present our graph edge-based semi-supervised learning approach.

Problem Definition: The aim of this paper is to find an optimum traffic light configuration in multi-intersections so that traffic congestion can be prevented as much as possible. Before we delve into formal approaches to solve this traffic management system (TMS) problem, let us first give some brief notations that will be used throughout the paper.

We define the TMS environment as $\mathcal{E}$, which consists of multiple intersections. Here, in this environment $\mathcal{E}$, each intersection is assumed to have an "intelligent" traffic light agent TL_i. TL_i and TL_j (for i ≠ j) in which "Red" and "Green" are used as the default phases settings. Moreover, when a light changes from one phase to another, an additional "yellow" phase is set for a given period of time (default time is 7s).

3.1. State-of-the-art-approaches

3.1.1. Occupancy based algorithm

The occupancy-based algorithm [37] is one of the heuristic algorithms that tries to solve the TMS problem by relying on the basic idea of using the predefined historical traffic data to determine occupancies of the roads. More specifically, in this algorithm, the occupancies of all roads with incoming traffic onto the intersection are taken into account and this data is then used to prioritize traffic phases, i.e if a road poses more occupancy, then its green phase duration is incremented up to a certain threshold. This prioritization formula can be given as follows:

$$D = T_{\max} * (N * (W_i/W_T)) \tag{1}$$

where D is the duration, which is also set to a certain limit to prevent blocking the other roads, $T_{\max}$ is the total time, N is the number of incoming roads, W_i is the occupancy value for incoming road i, which is heuristically set to a certain value by observing the historical traffic data, and W_T is the sum of all the road occupancy coming into the intersection.

Despite the effectiveness and simpleness of the occupancy algorithm, this algorithm sets its parameters based on historical data which is prone to change, i.e some roads may have higher occupancy values in the past, however, their occupancy value may decrease in the future. Thus, this algorithm introduces a certain bias towards the historically occupied roads.

3.1.2. Scoring algorithm

The scoring algorithm is another heuristic algorithm that is proposed by Sébastien Faye et al. [36]. The basic premise behind the scoring algorithm is to score each incoming road in an intersection and assign priorities based on these scores. Formally, these scores of each incoming road can be computed as follows:

$$LS = \alpha \cdot \left(\frac{N^{(s,d)}}{\sum_{\{a,b\} \in D} N^{(a,b)}} \right) + \beta \cdot \left(\frac{T_F^{(s,d)}}{\sigma_{\{a,b\} \in D} T_F^{(a,b)}} \right) \tag{2}$$

where α and β are user-defined weights to optimize average vehicle time and starvation, we use default values as 1 for both, (s, d) is a possible movement going from a direction s to another direction d within a set of all possible directions, $N^{(s,d)}$ is the weighted sum of the number of vehicles present on the coming lanes that compose movement, and $T_F^{(s,d)}$ is the last time since the incoming road had a green phase. In the case that there are no vehicles present on any of the roads, LS would be set to 0.

Finally, once the scores are obtained, the system calculates priority duration using the same equation as equation (1). Replacing occupancy with local score gives us:

$$D = T_{\max} * (N * (S_i / S_T)) \tag{3}$$

where D is the duration, $T_{\max}$ is the total time, N is the number of incoming roads. S_i and S_T represents the local score for incoming road i and the sum of all scores, respectively.

3.1.3. Reinforcement learning algorithms

In this paper, in addition to the heuristic-based approaches defined as above, we also employ two deep-learning algorithms for the TMS problem. More specifically, we use PPO [15] and A2C [14] trained using the Stable Baselines library [35]. While the classical DQN derives the policy by learning the Q-value function, these two policy-based algorithms improve the policy directly.

Proximal Policy Optimization (PPO) [15]: As given in formula (4), maximizing TRPO algorithm can lead to frequent parameter updates, hence instability. PPO is a policy gradient method, which simplifies TRPO by using a clipped surrogate objective. It forces $r(\theta)$ to fit into a smaller interval $[1 - \epsilon, 1 + \epsilon]$ using the $clip(r(\theta), 1 - \epsilon, 1 + \epsilon)$ function. In addition, PPO encourages exploration using an error term and an entropy term shown in the formula (5) where ϵ, $c1$, and $c2$ are hyper-parameters.

$$J^{\text{TRPO}}(\theta) = \mathbb{E}\left[r(\theta) \hat{A}_{\theta_{\text{old}}}(s, a) \right] \tag{4}$$

$$J^{\text{CLIP'}}(\theta) = \mathbb{E}\left[J^{\text{CLIP}}(\theta) - c_1 \left(V_\theta(s) - V_{\text{target}} \right)^2 + c_2 H\left(s, \pi_\theta(.)\right) \right] \tag{5}$$

Advantage Actor Critic (A2C) [14]: Actor critic methods use two separate networks: (i) actor and (ii) critic. The critic network learns the value function while the actor network is trained to update the policy according to the value function learned by the critic network. A2C makes use of the advantage function (6) in order to calculate the advantage of a specific action over average general action at a given state as follows:

$$A^\pi(s, a) = Q^\pi(s, a) - V^\pi(s) \tag{6}$$

For A2C [14] and PPO [15] algorithms, the reward is calculated based on the difference in waiting for the time of vehicles and the respective road movement as follows:

$$R = \begin{cases} -1 * \left(\sum T + \sum V \right) & \text{N} == 0 \\ -1 * \left(\frac{(\sum T + \sum V)}{N} \right) & \text{N} >= 1 \end{cases}$$

where R is the calculated reward, T is an array of waiting times for different vehicles, V is the number of new vehicles compared to the previous iteration, and N is the number of vehicles that moved to a different road.

3.2. Our proposed graph-semi supervised learning based approach

The basic idea behind our proposed approach is to predict which intersections are going to be occupied by vehicles by representing the TMS problem as a graph problem; and predicting the edge flow on this graph. More specifically, in our graph-based modeling, we treat each intersection as well as exit point as a node; and we treat the roads that are connecting these nodes as edges. More concretely, if we consider the 17 intersections network shown in Fig. 5, we can observe that there are an additional 12 sets of nodes that relate to different vehicle spawn/despawn points and exit points. Thus, in our graph-based representation, the graph has 29 nodes in total, which are marked with red dots in the figure. Any road that connects two distinct intersections is considered to be an edge between these nodes. In order to predict the traffic flow from one intersection (node) to another node connected via roads (edges), we use the methodology proposed in [12], which is an edge-based semi-supervised learning algorithm. This way, we aim at determining the green phase time for upcoming intersections. With the help of edge-based semi-supervised learning, each intersection's predicted inwards flow allows the intersection to optimize the total phase cycle according to the predicted traffic flows. In addition, the outward prediction of one intersection would be used as the inward prediction for the following intersections.

Mathematically, we have a set of vertices (traffic intersections and exit points) V, a set of edges that connect the edges (roads) ϵ, and a labeled set of edge flows (traffic flow) $\varepsilon \subseteq \epsilon$. The goal of the algorithm is to predict the unlabeled edge flows $\varepsilon^U \equiv \epsilon \backslash \varepsilon$. Here, it is important to note that our aim is to conduct edge based semi-supervised learning approach, rather than a well-known node-based semi-supervised learning method [38].

In **Graph-based Semi-Supervised Learning for vertices** [38], a graph is constructed with nodes and edges, where nodes are specified by labeled V^L and unlabeled samples V^U. Edges would be based on the similarities among the samples V. The goal of the algorithm would be to assign labels for the unlabeled samples V^U based on the existing data, such that the assigned labels vary smoothly across the neighboring nodes. The notion of smoothness can be defined by the log function as follows:

$$\| B^T y \|^2 = \sum_{(i,j) \in \epsilon} (y_i - y_j)^2 \tag{7}$$

where y represents the vector containing vertex labels, and $B \in \mathbf{R}^{nxm}$ represents the incidence matrix of the network. This loss function can be written as $\| B^T y \|^2 = y^T L y$ in terms of the graph Laplacian $L = BB^T$.

We can now obtain labels for the unknown nodes by minimizing the quadratic form $y^T L y$ with respect to y while keeping the labeled vertices fixed.

In the case of **Graph-based Semi-Supervised learning for edge flows** [12], we can represent the edge flows in the network with the vector f. If we account only for the $netflow$ along an edge, we obtain $f_r > 0$ when the flow orientation of edge r aligns with its reference orientation and $f_r < 0$ in all other cases. The true edge flow in the network is denoted by $\hat{f}$. The divergence of a vertex can be found by calculating the sum of outgoing flows minus the incoming flows at that vertex. This can be shown as follows:

$$(Bf)_i = \sum_{\varepsilon_r \in \varepsilon : \varepsilon_r \equiv (i,j), i<j} f_r - \sum_{\varepsilon_r \in \varepsilon : \varepsilon_r \equiv (j,i), j<i} f_r. \tag{8}$$

Additionally, we can also create a loss function for edge flows which enforces a notion of flow conservation. This can be expressed using the sum-of-squares vertex divergence:

$$\| Bf \|^2 = f^2 B^T B f = f^T L_e f \tag{9}$$

In the above equation; $L_e = B^T B$ is the edge Laplacian matrix.

In order to obtain the set of labeled edges that are most helpful in determining the overall edge flows in the network, we use **Active Semi-Supervised Learning**. Similar studies for vertex-based semi-supervised

learning can be found in [39,40]. Using active semi-supervision in our problem, we are able to determine which set of roads is most vital in decision-making. In terms of efficiency, we could use this information to deploy sensors only to this set of roads to gain optimal results and thus, reduce the algorithm costs and even the real-world deployment budget.

The approach we use is called **Rank-revealing QR (RRQR)** which is a heuristic method for the optimal column subset selection, which is an approximation to well-known maximum submatrix volume problem. This problem is an NP-hard problem [41]. The method proposes the idea that in order to select ε^L, we choose m^L rows from V_0 that would maximize the smallest singular value of the resulting submatrix. The RRQR heuristic computes:

$$V_C^T \Pi = Q[R_1 R_2]. \tag{10}$$

where Π is a permutation matrix that keeps R_1 well conditioned. Additionally, the first m^L columns of Π is chosen by the resulting edge set ε^L and edge indicated by the column permutation within Π.

In summary, we represent the TMS problem as a graph problem and aim at determining which intersections will be congested beforehand so that we can set more green times for those intersections that are predicted to be occupied via vehicles with the use of edge-based semi-supervised algorithm [12].

4. Results

In this section, we first give a brief background on the simulation environment that is used for evaluating the performance of algorithms used in this paper. We then explain the hyper-parameters in experimental setups. Finally, we give a comparative performance evaluation of the proposed algorithm against two deep learning-based algorithms, i.e., A2C [14] and PPO [15], as well as against two heuristic-based approaches [36] in terms of waiting time.

4.1. Simulation environment: Simulation of urban mobility (SUMO) platform

SUMO is an open-source traffic simulation suite that allows us to design, model, and simulate traffic networks [13]. During a simulation, SUMO provides the necessary tools to interact with the simulation.

We invoke various tools of SUMO, such as the NetEdit tool used to create and model various traffic networks. the Random Trips tool is used to create random vehicle trips, and the Duarouter tools allow us to generate vehicle routes based on a previously created network.

After obtaining a completed network model via the aforementioned tools of SUMO, we simulate the network using the graphical and command line interface tools. Interaction with an ongoing simulation is done using Traffic Control Interface (TraCI).

TraCI allows us to obtain information directly from an ongoing simulation. For the scope of this study, we extract network structure at the start of a simulation, and the occupancy of the relevant roads, each time a traffic phase algorithm is run.

In TraCI, vehicle occupancies are defined as the number of vehicles on a given road (edge). A given edge may have different number of lanes. The number of lanes within an edge would affect the total queue in the edge, but the vehicle occupancy remains the same. The size of the queue within an intersection can cause delays for the vehicles within the edge due to delays caused by initial movement from a stopping stage of the vehicle. In other words, a continuous flow of vehicles would be an ideal approach to reduce such delays. In this paper, all four-way intersections are designed as seen in Fig. 3, and three-way intersections are designed in a similar manner.

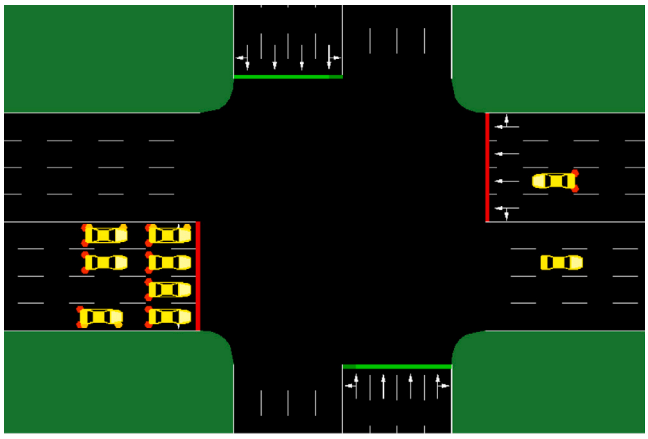

Fig. 3. Single Intersection network design.

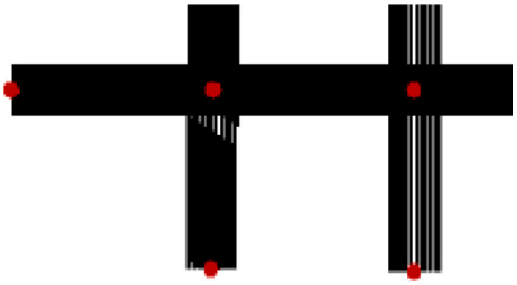

Fig. 4. Network design for simulation environment with 5 traffic light intersections.

4.2. Hyper-parameters

Network model and Datasets: We evaluate our proposed method on a synthetically generated network model using SUMO's NetEdit tool. The vehicles were generated using a combination of the RandomTrips tool and the Duarouter tool. We run the simulations with the following hyper-parameters:

- **Number of steps:** The number of steps determines how long a specific simulation runs. Here, ideally, we want all the generated vehicles to enter the simulation before the simulation ends. The time at which vehicles leave the simulation varies depending on which algorithms are used for traffic phase selection. In this paper, the number of step hyper-parameter is set to 1600.
- **Number of simulations:** The number of simulations determines how many times we run each algorithm. We run the algorithm multiple times and then take the average results obtained to fairly evaluate the performance of the algorithms. Thus, we repeat the simulations 10 times for each algorithm and report the average of the algorithms' performance.
- **Traffic Phases:** Each intersection in the network has predefined traffic phases. These phases are generated by the SUMO. We do not have any custom traffic phases, as our proposed method aims waiting time optimization only.

A sample network structure of a classic single intersection is shown in Fig. 3. Given that our approach focuses on multiple intersections, we integrate the proposed idea into different number of intersections and then connect them. An example of the final network for 5, 17 and 20 intersection networks can be seen in Fig. 4, 5 and 6. Additionally, not all intersections within any of the multi-intersection networks are the same, some of them may only intersect with three other roads (unlike the classic scenario where four other roads are present in an

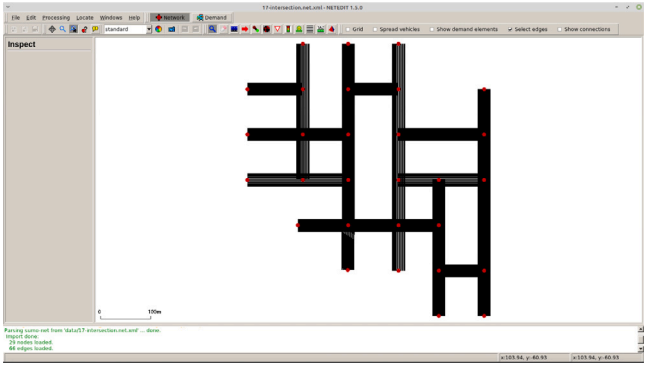

Fig. 5. Network design for simulation environment with 17 traffic light intersections.

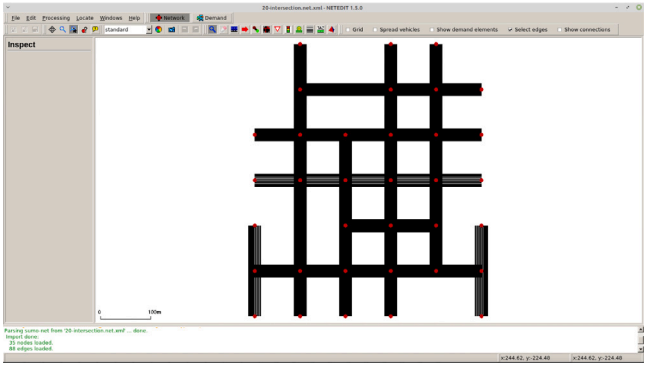

Fig. 6. Network design for simulation environment with 20 traffic light intersections.

intersection). This setup renders us a more versatile application of real-world road networks.

Preprocessing: In order to integrate the Flow Prediction algorithm, we pre-process some of the data obtained from the SUMO environment using TraCI.

At the start of a simulation using the Flow Prediction algorithm [12], we create additional files related to the network; a file to note the cartesian coordinates of every intersection in the network, and a secondary file noting all connections between intersections.

While running the simulation, we also use TraCI to obtain occupancy and occupancy change information on all the relevant roads. Each time we call the Flow Prediction algorithm, another file was generated noting all the occupancy change data with regard to the connection data previously generated.

Our experiments are conducted using Python3. The flow prediction algorithm is implemented in Julia Programming Language.

5. Discussion on performance evaluations

To assess the performance of the proposed algorithm against two deep learning-based algorithms A2C [14] and PPO [15] as well as two heuristic-based approaches [36], we use average wait time, which is defined as vehicles' average wait time during the simulation and a commonly used performance metric in the related literature. To this end, we create three different intersection configurations, namely 5, 17, and 20 intersections in the SUMO platform using its TraCI tool. The traffic routes are randomly generated each time the simulation is run.

Figs. 7 and 8 show the performance evaluation in the average total waiting time and the average maximum waiting time for a vehicle in a simulation. These results are gained by running the simulation 10 times, generating new vehicle routes at each time. Specifically,

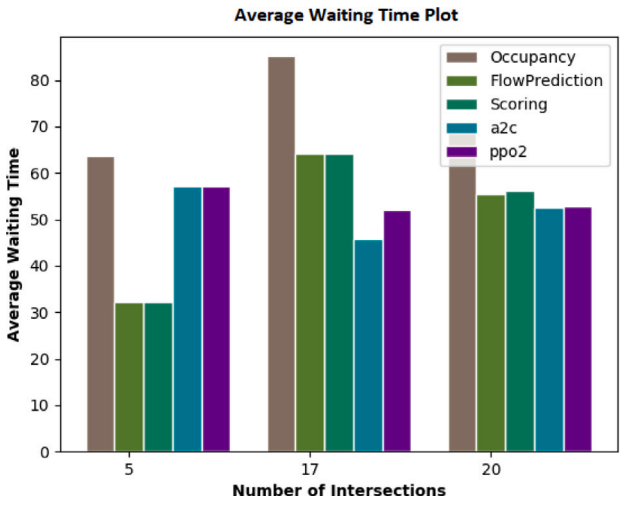

Fig. 7. Average wait time obtained when simulating network models with 5, 17 and 20 intersections.

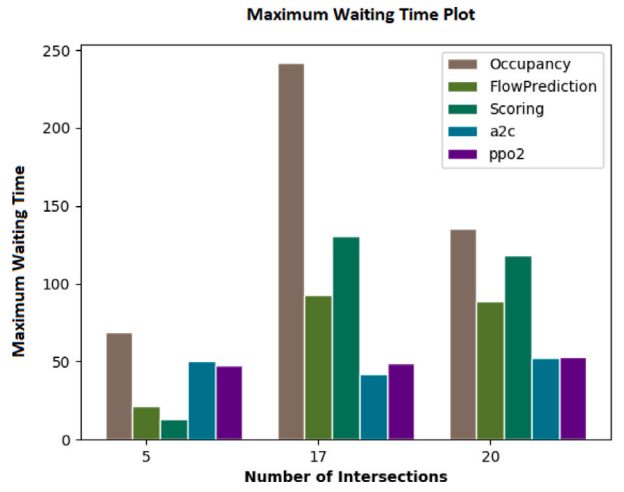

Fig. 8. Maximum wait time obtained when simulating network models with 5, 17 and 20 intersections.

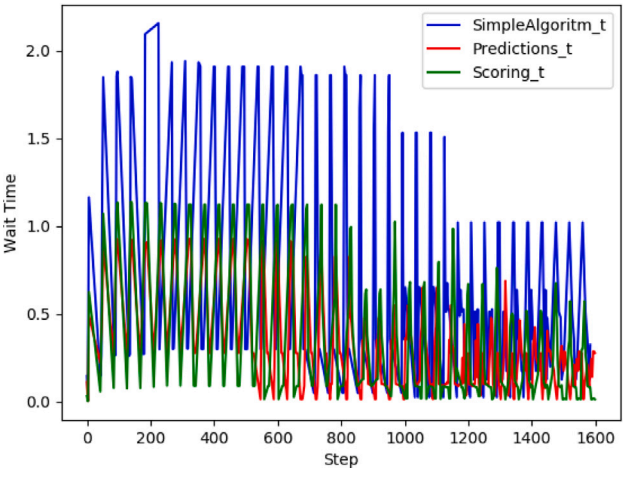

Fig. 9. Results obtained for simulation with 1600 steps. The simulation was performed 10 times and the average wait times were taken.

the simulations are conducted over a period of 1600 steps and each algorithm runs a total of 10 times; around every 160 steps.

Table 2
Average and maximum waiting time based evaluations' summarization table.

Algorithms	Average waiting time	Maximum waiting time	Number of intersections
Occupancy-based Alg.	63	69	5
Flow-based Alg. (Ours)	32	19	5
Scoring-based Alg.	32	15	5
A2C [14]	57	48	5
PPO2 [15]	57	45	5
Occupancy-based Alg.	85	241	17
Flow-based Alg.	64	91	17
Scoring-based Alg.	64	126	17
A2C [14]	45	39	17
PP02 [15]	52	46	17
Occupancy-based Alg.	68	131	20
Flow-based Alg.	55	88	20
Scoring-based Alg.	56	118	20
A2C [14]	52	49	20
PP02 [15]	53	50	20

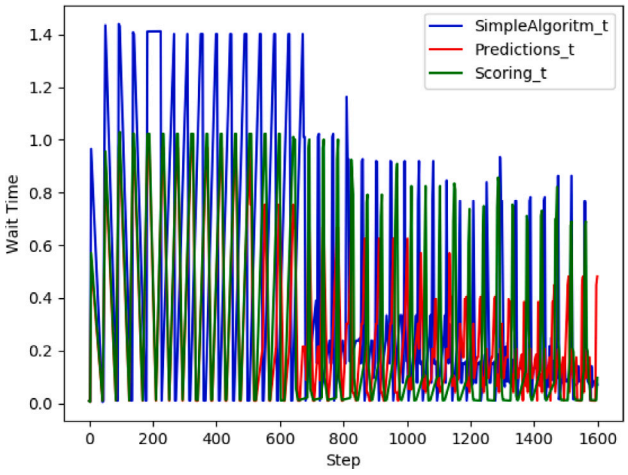

Fig. 10. Results obtained for simulation with 20 traffic light intersections, 1600 steps for each simulation. The simulation was performed 10 times and the average wait times were taken.

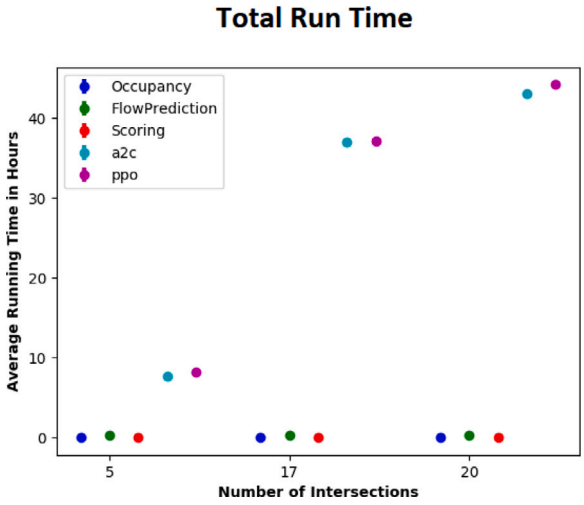

Fig. 11. Run time (in hours) for each algorithm in the network model with 5, 17 and 20 intersections.

To better highlight Figs. 7 and 8, we also present Table 2 that summarizes the average and maximum waiting times for each vehicle in the simulation. As it can seen in the table, our proposed approach delivers comparable results to the baseline algorithms in terms of average and maximum waiting time with much less running time, see Fig. 11.

Figs. 9 and 10 show the average waiting time for a single vehicle against the time step in a 17 intersection and a 20 intersection simulation, values shown in the graph are average from 10 distinct simulation runs. From the figures, we can observe that our algorithm (Flow-based algorithm) can deliver better results than that of heuristic algorithms and comparable results to that of deep learning algorithms.

Specifically, we also compare the runtime performances of the algorithms in Fig. 11. It can be clearly seen that the runtime of our algorithm is in minutes while the runtime may take several days for the deep learning algorithms. This finding suggests that the proposed algorithm is more practical in real-world traffic datasets compared with deep learning algorithms, i.e., A2C [14] and PPO2 [15].

Finally, we also examine the effects of known edge-label in our algorithm. To this end, we gradually increase the known edge labels and observe their effect. Fig. 12 shows the performance of our algorithm in all of the different network models, compared with the ZeroFill algorithm in terms of prediction. We see that the algorithm performs better when the number of intersections increases, indicating that graph topological information is useful.

The main findings of the performance evaluation can be summarized as follows:

- The proposed flow prediction algorithm outperforms other algorithms, since it can predict where traffic congestion will occur.
- Having larger number of nodes (intersections) provides more accurate results in terms of vehicle flow prediction. Large number of intersections are more informative, and thus, prediction accuracy is improved.
- To the best of our knowledge, no other study uses edge-based semi-supervised learning to predict vehicle flows between traffic intersections.

6. Conclusion

In this paper, we propose an adaptive traffic light management system that is able to predict traffic flow from one intersection to another. The principal algorithm behind the proposed system is graph-based semi-supervised learning for edge flows, where each traffic light intersection, vehicle spawn and despawn points are considered as a vertex; and the roads connecting any two vertices are considered as an edge. Magnitudes of edge connections are then calculated using the proposed RRQR method. The obtained information is then used to select and optimize the predefined traffic phases.

Future efforts in this direction would include further optimization of the traffic selection system itself. Comparative performance evaluations on various traffic intersection configurations show that our approach can produce comparable average vehicle waiting time and drastically

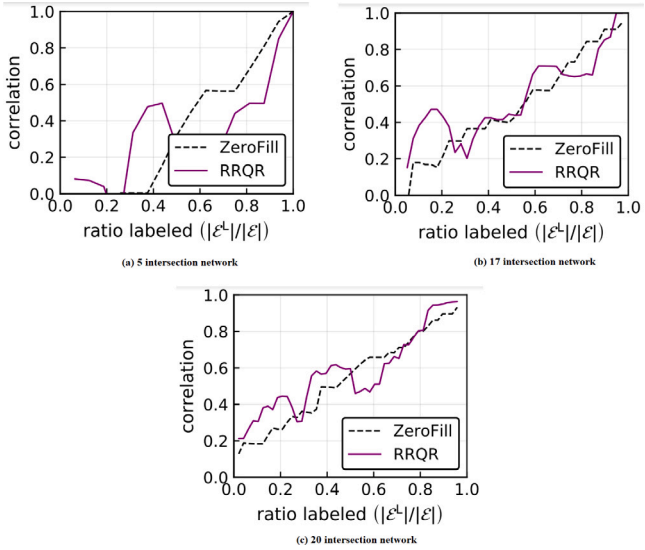

Fig. 12. Graph based SSL for synthetic flows from the simulation on 5, 17 and 20 intersection network models. The plots show the Pearson correlation between the estimated flow f^* and the ground truth flow $\hat{f}$ as a function of the ratio of labeled edges.

reduce the training/learning time of learning adequate traffic light configurations for all intersections within a few seconds, while deep learning-based approach can be trained in a few days for learning similar light configurations.

Declaration of competing interest

This paper has not been published in any other journal.

Data availability

Data will be made available on request.

Acknowledgments

This work was supported by the Turkish Scientific and Technical Research Council (TUBITAK) TEYDEB Program under Project No: 3220798 and produced from the master thesis [42].

References

[1] INRIX, Home.
[2] Alan J. Miller, Settings for fixed-cycle traffic signals, J. Oper. Res. Soc. 14 (4) (1963) 373–386.
[3] Fo Vo Webster, Traffic Signal Settings, Technical Rreport, 1958.
[4] Seung-Bae Cools, Carlos Gershenson, Bart D'Hooghe, Self-organizing traffic lights: A realistic simulation, in: Advances in Applied Self-Organizing Systems, Springer, 2013, pp. 45–55.
[5] M. Coşkun, A. Baggag, S. Chawla, Deep reinforcement learning for traffic light optimization, in: 2018 IEEE International Conference on Data Mining Workshops, ICDMW, 2018, pp. 564–571.
[6] Lior Kuyer, Shimon Whiteson, Bram Bakker, Nikos Vlassis, Multiagent reinforcement learning for urban traffic control using coordination graphs, in: Joint European Conference on Machine Learning and Knowledge Discovery in Databases, Springer, 2008, pp. 656–671.
[7] M.A. Wiering, Multi-agent reinforcement learning for traffic light control, in: Machine Learning: Proceedings of the Seventeenth International Conference, ICML'2000, 2000, pp. 1151–1158.
[8] Elise Van der Pol, Frans A. Oliehoek, Coordinated deep reinforcement learners for traffic light control, Proc. Learn., Inference Control Multi-Agent Syst., (At NIPS 2016) (2016).
[9] Hua Wei, Guanjie Zheng, Vikash Gayah, Zhenhui Li, Recent advances in reinforcement learning for traffic signal control: A survey of models and evaluation, ACM SIGKDD Explor. Newsl. 22 (2) (2021) 12–18.
[10] Hua Wei, Nan Xu, Huichu Zhang, Guanjie Zheng, Xinshi Zang, Chacha Chen, Weinan Zhang, Yanmin Zhu, Kai Xu, Zhenhui Li, Colight: Learning network-level cooperation for traffic signal control, in: Proceedings of the 28th ACM International Conference on Information and Knowledge Management, 2019, pp. 1913–1922.
[11] Hua Wei, Guanjie Zheng, Huaxiu Yao, Zhenhui Li, Intellilight: A reinforcement learning approach for intelligent traffic light control, in: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, ACM, 2018, pp. 2496–2505.
[12] Junteng Jia, Michael T. Schaub, Santiago Segarra, Austin R. Benson, Graph-based semi-supervised & active learning for edge flows, 2019, arXiv preprint arXiv:1905.07451.
[13] Pablo Alvarez Lopez, Michael Behrisch, Laura Bieker-Walz, Jakob Erdmann, Yun-Pang Flötteröd, Robert Hilbrich, Leonhard Lücken, Johannes Rummel, Peter Wagner, Evamarie Wießner, Microscopic traffic simulation using SUMO, in: The 21st IEEE International Conference on Intelligent Transportation Systems, IEEE, 2018.
[14] Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Timothy P. Lillicrap, Tim Harley, David Silver, Koray Kavukcuoglu, Asynchronous methods for deep reinforcement learning, 2016, arXiv:1602.01783.
[15] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, Oleg Klimov, Proximal policy optimization algorithms, 2017.
[16] Francois Dion, Hesham Rakha, Youn-Soo Kang, Comparison of delay estimates at under-saturated and over-saturated pre-timed signalized intersections, Transp. Res. B 38 (2) (2004) 99–122.
[17] Brian L. Smith, Michael J. Demetsky, Short-term traffic flow prediction: Neural network approach, Transp. Res. Rec. (1453) (1994).
[18] Guanjie Zheng, Yuanhao Xiong, Xinshi Zang, Jie Feng, Hua Wei, Huichu Zhang, Yong Li, Kai Xu, Zhenhui Li, Learning phase competition for traffic signal control, 2019, arXiv preprint arXiv:1905.04722.
[19] Hsin-Hung Pan, Shu-Ching Wang, Kuo-Qin Yan, An integrated data exchange platform for intelligent transportation systems, Comput. Stand. Interfaces 36 (3) (2014) 657–671, URL https://www.sciencedirect.com/science/article/pii/S0920548913000949.
[20] S.L. Toral, F. Barrero, F. Cortés, D. Gregor, Analysis of embedded CORBA middleware performance on urban distributed transportation equipments, Comput. Stand. Interfaces 35 (1) (2013) 150–157.
[21] D. Gregor, S. Toral, T. Ariza, F. Barrero, R. Gregor, J. Rodas, M. Arzamendia, A methodology for structured ontology construction applied to intelligent transportation systems, Comput. Stand. Interfaces 47 (2016) 108–119.
[22] Federico Barrero, Jean A. Guevara, Enrique Vargas, Sergio Toral, Manuel Vargas, Networked transducers in intelligent transportation systems based on the IEEE 1451 standard, Comput. Stand. Interfaces 36 (2) (2014) 300–311.
[23] I. Román, G. Madinabeitia, L. Jimenez, G.A. Molina, J.A. Ternero, Experiences applying RM-ODP principles and techniques to intelligent transportation system architectures, Comput. Stand. Interfaces 35 (3) (2013) 338–347, RM-ODP: Foundations, Experience and Applications.
[24] Yoichiro Iwasaki, Image processing system to measure vehicular queues and an adaptive traffic signal control by using the information of the queues, Comput. Stand. Interfaces 20 (6) (1999) 444.
[25] Waris Hooda, Pradeep Kumar Yadav, Amogh Bhole, Deptii D. Chaudhari, An image processing approach to intelligent traffic management system, in: Proceedings of the Second International Conference on Information and Communication Technology for Competitive Strategies, ICTCS '16, Association for Computing Machinery, New York, NY, USA, 2016.
[26] Ninad Lanke, Sheetal Koul, Smart traffic management system, Int. J. Comput. Appl. 75 (7) (2013).
[27] S. Sheik Mohammed Ali, Boby George, Lelitha Vanajakshi, Jayashankar Venkataraman, A multiple inductive loop vehicle detection system for heterogeneous and lane-less traffic, IEEE Trans. Instrum. Meas. 61 (5) (2011) 1353–1360.
[28] Isaac Porche, Stéphane Lafortune, Adaptive look-ahead optimization of traffic signals, J. Intell. Transp. Syst. 4 (3–4) (1999) 209–254.
[29] Baher Abdulhai, Rob Pringle, Grigoris J. Karakoulas, Reinforcement learning for true adaptive traffic signal control, J. Transp. Eng. 129 (3) (2003) 278–285.
[30] Yit Kwong Chin, Nurmin Bolong, Aroland Kiring, Soo Siang Yang, Kenneth Tze Kin Teo, Q-learning based traffic optimization in management of signal timing plan, Int. J. Simul., Syst., Sci. Technol. 12 (3) (2011) 29–35.
[31] Patrick Mannion, Jim Duggan, Enda Howley, An experimental review of reinforcement learning algorithms for adaptive traffic signal control, in: Autonomic Road Transport Support Systems, Springer, 2016, pp. 47–66.
[32] P.G. Balaji, X. German, Dipti Srinivasan, Urban traffic signal control using reinforcement learning agents, IET Intell. Transp. Syst. 4 (3) (2010) 177–188.
[33] Itamar Arel, Cong Liu, Tom Urbanik, Airton G. Kohls, Reinforcement learning-based multi-agent system for network traffic signal control, IET Intell. Transp. Syst. 4 (2) (2010) 128–135.
[34] Samah El-Tantawy, Baher Abdulhai, Hossam Abdelgawad, Multiagent reinforcement learning for integrated network of adaptive traffic signal controllers (MARLIN-ATSC): Methodology and large-scale application on downtown toronto, IEEE Trans. Intell. Transp. Syst. 14 (3) (2013) 1140–1150.

[35] Ashley Hill, Antonin Raffin, Maximilian Ernestus, Adam Gleave, Anssi Kanervisto, Rene Traore, Prafulla Dhariwal, Christopher Hesse, Oleg Klimov, Alex Nichol, Matthias Plappert, Alec Radford, John Schulman, Szymon Sidor, Yuhuai Wu, Stable baselines, 2018, https://github.com/hill-a/stable-baselines.

[36] Sébastien Faye, Claude Chaudet, Isabelle Demeure, A distributed algorithm for multiple intersections adaptive traffic lights control using a wireless sensor networks, in: Proceedings of the First Workshop on Urban Networking, UrbaNe '12, Association for Computing Machinery, New York, NY, USA, 2012, pp. 13–18.

[37] Sultan Kübra Can, Adam Thahir, Mustafa Coşkun, V. Çağrı Güngör, Traffic light management systems using reinforcement learning, in: 2022 Innovations in Intelligent Systems and Applications Conference, ASYU, IEEE, 2022, pp. 1–6.

[38] Mustafa Coskun, Burcu Bakir Gungor, Mehmet Koyuturk, Expanding label sets for graph convolutional networks, 2019, arXiv preprint arXiv:1912.09575.

[39] Akshay Gadde, Aamir Anis, Antonio Ortega, Active semi-supervised learning using sampling theory for graph signals, in: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2014, pp. 492–501.

[40] Andrew Guillory, Jeff A. Bilmes, Label selection on graphs, in: Advances in Neural Information Processing Systems, 2009, pp. 691–699.

[41] Ali Çivril, Malik Magdon-Ismail, On selecting a maximum volume sub-matrix of a matrix and related problems, Theoret. Comput. Sci. 410 (47–49) (2009) 4801–4811.

[42] Adam Rizvi Thahir, Graph Theory Based Traffic Light Management (Master's thesis), Abdullah Gül Üniversitesi, Fen Bilimleri Enstitüsü, 2022.