Knowledge Engineering and Data Science (KEDS)
Vol 4, No 2, December 2021, pp. 85–96

pISSN 2597-4602
eISSN 2597-4637

A Comprehensive Analysis of Reward Function for Adaptive Traffic Signal Control

Abu Rafe Md Jamil [1], Naushin Nower [2], *

Institute of Information Technology, University of Dhaka
Dhaka-1000, Bangladesh

[1] *bsse0722@iit.du.ac.bd;* [2] *naushin@iit.du.ac.bd**
** corresponding author*

ARTICLE INFO

Article history:
Submitted 14 October 2021
Revised 9 November 2021
Accepted 19 November 2021
Published online 31 December 2021

Keywords:
Reward function
Adaptive traffic signal control
Deep reinforcement learning
Multi-objective reward function
Signal Timing

ABSTRACT

Adaptive traffic control systems (ATCS) can play an essential role in reducing traffic congestion in urban areas. The main challenge for ATSC is to determine the proper signal timing. Recently, Deep Reinforcement Learning (DRL) has been used to determine proper signal timing. However, the success of the DRL algorithm depends on the appropriate reward function design. There exist various reward functions for ATSC in the existing research. This research presents a comprehensive analysis of the widely used reward function. The pros and cons of various reward algorithms were discussed, and experimental analysis shows that the multi-objective reward function enhances the performance of ATSC.

I. Introduction

Modern society primarily depends on public road transport for their movement and goods. Thus, the transportation system is a critical infrastructure for the metropolitan city since it plays a vital role in the habitant's daily life and environment. Moreover, the increasing population requires an increasing number of vehicles on the road, often exceeding the infrastructure's total capability, leading to more congestion and longer travel time. Because of horrible traffic congestion, three billion gallons of fuel are wasted each year, and travelers need to wait in their cars for almost seven billion extra hours- (forty-two hours per traveler) [1]. Thus, traffic jam has become a severe problem in almost every metropolitan area because of population growth, increasing urbanization, inadequate traffic infrastructure, and inefficient traffic signal control. However, in the era of modern civilization, it is pretty impossible to stop urbanization; as a result, more public transportation is needed to facilitate the increasing population. Increasing the transport capability may be a solution to this issue through road infrastructure construction. However, it is costly and time-consuming. Whereas enhancing the issue of traffic signal control could be an effective solution to this problem. Reducing 1% traffic congestion using an efficient traffic signal control system can save billions in a year [2].

Among the various traffic control systems, the adaptive traffic signal control (ATSC) system is the most appropriate solution to the traffic signal control problem since it utilizes real-time traffic information from road intersections to make the signal control decision. Unlike fixed signal timing, it makes dynamic traffic signaling based on the incoming data [3] more suitable for reducing traffic jams. ATSC catches the attention of researchers because it can intelligently regulate the traffic signal to minimize traffic congestion. The traffic controller is needed to be correctly trained to make the proper signal timing to mitigate congestion. Recently Deep Reinforcement Learning (DRL) has been used to train the traffic controller [4][5][6][7] in many kinds of research. In the traffic signal control problem, the proper decision is unknown, and to evaluate the decision, we need to wait until the effect

https://doi.org/10.17977/um018v4i22021p85-96

of the decision is observed. This type of problem can be solved by Deep Q-Learning Network (DQN), a type of DRL.

In DRL, the traffic light controller observes the present traffic condition as a state, and it has a decision unit that decides which traffic phase to activate in a particular situation, i.e., which traffic light turns green. According to the decision of the controller, the action is employed. The traffic condition changes after the action and is based on the traffic condition, and a scalar reward value is gained from the environment shown in Figure 1. In DRL, the impact of the decision is evaluated by reward value. If the reward value is high, the decision is labeled as good and bad if the reward value is low. The traffic light controller is trained to make decisions in different traffic conditions to gain high reward values to change traffic phases effectively to improve the traffic flow. Thus, reward value plays an essential role in DRL, and it can act as an evaluator of the action. The main objective of the DRL agent is to maximize the reward collected from the environment over time. More specifically, the reward function describes the problem the agent is trying to solve. It also defines the best possible performance. Therefore, the reward function is vital in how the agent learns optimal behavior [8]. Thus, selecting the reward function is one of the essential aspects of DRL for ATSC.

Different traffic features have been utilized to define the reward function in the existing DQN solutions [8]. Based on how many traffic features are utilized to design the reward function, the reward function can be classified as single objective and multi-objective. The reward function optimizes a single traffic feature is a single objective reward function. For example, the queue length [5][9], cumulative delay [7], waiting time [4][10], and travel time. Parameters are widely used as a single objective reward function. The work in [11] used vehicle pressure of the road intersection as a single objective reward function. Besides the single objective reward function, the multi-objective reward function optimizes more than one traffic parameter at a time. In this type of reward function, various traffic features are combined in different ways to optimize parameters.

Since most of the problems in the real world need multiple parameters to optimize, multi-objective reward functions are more appropriate than single objective reward functions. Many existing studies [12][13] also experimentally prove this fact. However, when conflicting parameters are combined in the multi-objective reward function, it does not optimize multiple parameters properly because of correlation among them. In order to solve this problem, in [8], the authors proposed composite reward architecture (CRA), where each reward function is evaluated separately, and a decision is made based on the majority's decisions. Before that, hybrid reward architecture [14] decomposes a single reward function into *n* different reward functions, and the summation of the *n* rewards calculates the final reward. Different types of reward functions have been proposed in the literature, and none of them compares among these reward functions. This research uses six single objective reward functions and three recently proposed multi-objective reward functions in different traffic scenarios. The comparative analysis can conclude that a multi-objective reward function is better than a single objective reward and among the multi-objective reward function, CRA [8] and [6] perform competitively.

II. Methods

Reinforcement Learning (RL) is a machine learning technique that can be learned through experience. RL is different from supervised learning [15], which learns from a labeled dataset with a supervisor. Supervised learning learns from a training set of labeled samples provided by a knowledgeable external supervisor. RL is also different from unsupervised learning [16], which is about learning clustering of unlabeled data. On the other hand, RL tries to optimize a reward signal by trial and error. Finding structure in an agent's experience may be helpful for RL, but unsupervised learning does not solve the RL issue of optimizing a reward signal. Thus, RL is the learning process of mapping scenarios to actions to maximize a numeric reward value.

In RL, an agent is deployed in an environment without any experience of behaving. The agent perceives the environment, determines an action based on the environment condition, and gets a reward from the environment based on the outcome of the action. Thus, the reward value is considered the environment's feedback for the action taken. If the environment's condition improves, the agent gets a positive reward; otherwise, it receives a negative reward. This way, the agent can distinguish between good and bad actions and gain experience iteratively. The expected reward value for each state-action pair is represented as a Q value and stored in a Q table. The agent can quickly determine which action is desirable for a particular state from this Q-table.

Any RL problem can be modeled as a Markov Decision Process (MDP), a mathematical framework to describe the environment in the RL. The MDP can be represented as a tuple $<S,A,T,R.>$, where S is the set of states, A represents a set of actions, T indicates a transition function, and R is the reward function. In any given state $s \in S$, selecting an action, $a \in A$ will change the environment to a new state $s' \in S$ with the probability $T(s,a,s') \in 0,1)$ and the environment gives a reward $r = R(s,a)$.

This process iteratively rewards the agent, generating a policy that can map certain states to action (π:S×A →[0,1]. An MDP aims to find the best policy for an agent. The policy π defines the probability of selecting action a_t in-state s_t. The policy is generated to maximize the expected cumulative discounted reward over time. The discounted future reward, R_t at time t is defined as follows

$$R_t = E[\sum_{k=0}^{\infty} \gamma^k r_{t+k}] \tag{1}$$

where γ is the discount factor which indicates the impact of future rewards.

In ATSC, there are a considerable number of state and action pairs. As a result, it is pretty impossible to manage them in a Q-table from [17]. This problem can be solved by using Deep Neural Network (DNN) where neural networks have been used for function approximation instead of Q-table. This type of function approximator is widely used in many studies where the state-action pair is unlimited and unknown. The neural network (NN) consists of many neurons with weights that can be easily used for unlimited state-action pairs. The problem of ATSC formulated with DRL is shown in Figure 1 where the road traffic scenario represents the environment. Traffic conditions such as queue length, waiting time, halting number, etc., are used to represent the state. The traffic controller acts as an agent. The neural network (in the agent) takes the values of the state from the environment as input and produces Q values. Based on the Q-values, the action is generated by the agent. The condition of the environment changes, and the agent gets a numerical reward value as the evaluation of the action. In this way, the agent learns the good or bad action and adjusts the weight of the network accordingly.

The state is the agent's view of the environment in a specific timestep. In literature, the state is designed in several aspects. It is better to design a state with more information to represent the current condition of the traffic. In the literature, some approaches use one parameter as a state; others use multiple parameters to represent the environment. Queue lengths of the lane, number of halting vehicles, waiting time of the vehicles, the average speed of the vehicles are used to represent state [6][7][8].

The action indicates the activity the agent performs in the environment. In the case of ATSC, the action is to change the traffic phase or stay in the current phase. The traffic phase could be two, four, or six. For two phases of traffic, vehicles going east and west across the intersection indicate one phase, and vehicles approaching north and south represent another phase. The traffic phase will be green for vehicles heading east and west, and in another action, the traffic light will be green for vehicles approaching north and south. As a result, the agent has two distinct actions $(A = \{0, 1\})$. Here, 0 instructs to stay on the current phase, and 1 indicates to change the traffic phase to the next phase. With these two actions, any phase of traffic can be controlled.

In RL, the reward reflects the environmental feedback after the agent has taken action. The agent uses the reward value to consider the result of the action taken and update the model for future choice of action. Therefore, the reward is one of the most vital parts of the learning process. Usually, it is

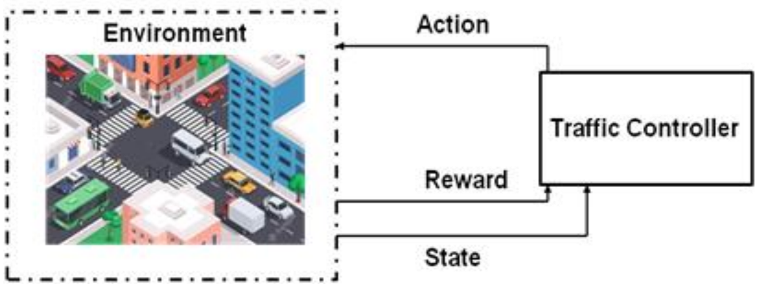

Fig. 1. The formulation of ATSC with DRL

defined as a function of some performance metrics of the intersection, such as vehicle delay, queue length, waiting time, travel time, or throughput. There are multiple ways to define reward functions. As a result, various reward architectures are designed in the literature, but none compare among different reward functions. In this paper, we have discussed the pros and cons of various existing reward functions and made an experimental analysis of widely used rewards on different traffic scenarios.

To make a comparative analysis among rewards, we need to focus on the types of rewards. In this section, the classification of reward functions is discussed. Based on the number of parameters and how the parameters are processed, the reward functions in the existing literature are designed basically in two ways: 1) Single objective and 2) Multi-objective.

The *Single objective reward function* is designed to optimize one parameter is known as a single objective reward function. In this approach, the agent's goal is to optimize the single parameter and get a reward based on how well it optimizes that one parameter. For example, in ATSC, queue length, delay, waiting time, travel time, etc., can be used separately as a traffic parameter to optimize. The single objective RL is depicted in Figure 2.

The work in [5] used DRL with a DNN to learn the Q-function. A deep stacked autoencoder (SAE) neural network [18] is used to estimate the Q-function. The SAE is trained to minimize the loss that indicates the difference of target and prediction Q-values in the learning. The reward function is designed considering only the queue lengths of the lanes.

$$r_t = \left| max_{i=1,2}\{q_t^{e-w,i}, q_t^{w-e,i}\} - max_{i=1,2}\{q_t^{s-n,i}, q_t^{n-s,i}\}\right| \quad (2)$$

where *i* indicates the lane number, *e-w* is east to west direction, *w-e* is west to east direction, *n-s* is north to south direction, *s-n* is south to north direction, and q_t is the queue length at time *t*. with the reward function in equation 2, the learning process maximizes queue length between we and ns. if the difference is high, the agent gets a high reward value. However, the difference in queue length between we and ns does not indicate the smoothness of traffic flow. For example, if we's queue length is 100 units and sn is 20 units, the reward value will be 80.

On the other hand, if the queue length in we is 40 units and sn is 40 units, the reward value will be zero. Although the second scenario represents a more stable traffic flow, it gets a meager reward. Therefore, the reward function gives wrong feedback to the agent.

Another atsc applying drl is proposed in [7]. This work proposed a dense information new state space with the discrete encoding of traffic features defined as discrete traffic state encoding (dtse). A deep convolution neural network trains the q-learning agent with experience replay [19]. The difference of cumulative delays (equation (3)) is considered as a reward function in this work.

$$r_t = d_{t-1} - d_t \quad (3)$$

where d_{t-1} is the cumulative delay at time $t - 1$, and d_t is at *t*. delay is defined in equation (4).

$$D = 1 - \frac{\text{average speed of vehicles in lanes}}{\text{maximum allowed lane speed}} \quad (4)$$

With this reward function, the agent tends to maximize the maximum possible speed and average speed of vehicles. The average speed of vehicles is influenced by road infrastructure, road occupancy, etc. the high average speed 24 of vehicles can be a metric for smooth traffic flow. However, it alone cannot guarantee the smoothness of traffic flow.

Further, another drl algorithm for atsc is proposed in [4]. The work used experience replay and target network mechanisms to improve algorithm stability.

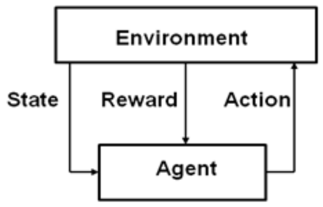

Fig. 2. Single objective rl

The difference of vehicles' cumulative waiting time between before and after the action, as shown in equation (5), is considered a reward function in [4]. this same reward function is used in another drl-based approach proposed in [10].

$$r_t = w_{t-1} - w_t \tag{5}$$

where w_{t-1} is the waiting time of vehicles at time $t-1$, and w_t is the waiting time at time t. with this reward function, the agent tries to minimize the waiting time for vehicles. Although waiting time is one of the most dominant parameters for indicating smooth traffic flow, only the waiting time of vehicles cannot provide an appropriate measure for efficient traffic flow. For example, if one vehicle has a low speed, the waiting time of the vehicle will be zero (waiting time is calculated only for the halting vehicles). Although the waiting time is zero, the vehicle is not experiencing a smooth journey. The above analysis can conclude that a single parameter alone cannot guarantee good performance. Thus, the traffic signal control problem is a multi-objective problem, where multiple parameters need to be optimized for better traffic flow. The single objective reward function optimizes a single parameter that is unsuitable for the traffic signal problem. Therefore, the multi-objective reward function is preferable for the atsc.

A *multi-objective* reward function is called a multi-objective reward function when it is designed to optimize multiple parameters at a time. In the real world, most problems are multi-objective; thus, this reward function is more suitable than a single objective reward function. In this approach, multiple parameters are combined to optimize the reward function. The learning agent optimizes the set of multiple objectives simultaneously. The multi-objective rl is depicted in Figure 3. In traffic signal control, queue length, delay, waiting time, travel time, and fuel consumption. Parameters could be combined with different weights for multi-objective optimization. The general form of the multi-objective reward function can be expressed in equation (6), which is the weighted sum of the traffic features.

$$r_t = \sum_{i=1}^{n} W_i \times tf_i \tag{6}$$

where W_i is the weight of i the traffic features and tf_i is the traffic feature.

The work in [19] investigated the learning control policies for traffic lights. It introduced a new reward function that considers the number of teleports j, number of action switches c, number of emergencies stop e, a sum of delay d, and a sum of wait time w as parameters.

$$r_t = -0.1c - 0.1\sum_{i=1}^{N} j_i - 0.2\sum_{i=1}^{N} e_i - 0.3\sum_{i=1}^{N} d_i - 0.3\sum_{i=1}^{N} w_i \tag{7}$$

The first three coefficients of the equation do not affect the DRL process [20]. For this reason, the feedback returned from the reward function may be misled.

A DRL agent with a multi-objective reward function is proposed in [6]. The research used a new DNN to decide whether to change the current traffic phase or keep measuring the Q-value. The neural network has different branches for each traffic phase. One branch of the network is activated in one traffic phase, and the other is activated in another traffic phase.

The reward function is calculated considering the weighted sum of the sum of delay D, queue length L, the sum of updated waiting time W, the total number of vehicles passed N, an indicator of light switches C, and total travel time T of the passing vehicles.

$$r_t = -0.25\sum_{i\in l} L_i - 0.25\sum_{i\in l} D_i - 0.25\sum_{i\in l} W_i - 5C + N + T \tag{8}$$

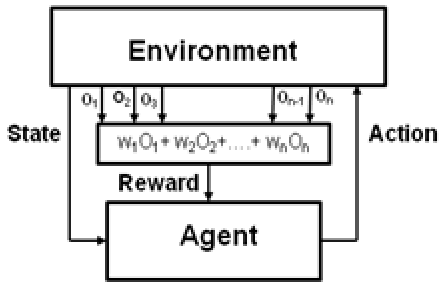

Fig. 3. Multi-objective RL

Here in equation (8), the indicator of light switches *C* has a very high negative coefficient though it does not affect the DRL process [20]. Furthermore, the total travel time *T* is added to the equation. It indicates that the agent will get a high reward whenever the travel time increases. However, a higher value of travel time indicates terrible traffic conditions. These two factors mislead the feedback with this reward function.

Further, they tested both on simulation data and data from the real world. This model achieves state-of-the-art efficiency on most measures. Nonetheless, it should be noted that the authors point out that due to the large volume of data collected for the agent training, the study has substantial limitations on a real-world appliance.

A new multi-objective reward function for traffic light optimization is proposed in [21]. It used deep Q-learning with a policy gradient approach to solve the RL problem. The following equation designs the reward function

$$r_t = D_{t-1} - D_t + \frac{Occupancy}{Number\ of\ Halting\ vechiles + c} \tag{9}$$

Here, the value is used to avoid zero division. Further, other multi-objective reward functions are proposed in [22][23][24]. Most reward functions are designed using the weighted sum of traffic features (equation 6). However, if the traffic features are conflicting and/or correlated with each other, the multi-objective reward function calculated by combining traffic features does not provide an optimal solution [12][13]. In order to solve this problem, the CRA [8] calculates the reward function for each traffic parameter separately and then combines the decisions of the multiple reward function by using the majority voting approach. In this multi-objective approach, each objective has a reward function, producing a decision. The decision chosen by the majority reward functions is selected from the multiple decisions. Since the reward functions are calculated separately, the dependency or confliction does not hamper making final decisions. However, this approach ignores the decisions of minority groups. The CRA approach is depicted in Figure 4.

HRA [14] is a particular type of reward function, which is a different form of the single and multi-objective reward function. In HRA [14], a task is divided into multiple tasks, and there is a reward function for each task. The final reward is calculated by adding the rewards of all sub-tasks. The HRA [14] is depicted in Figure 5.

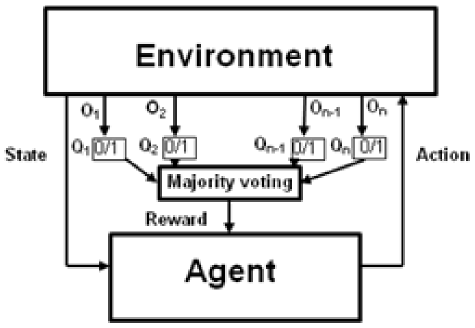

Fig. 4. Composite reward architecture for RL

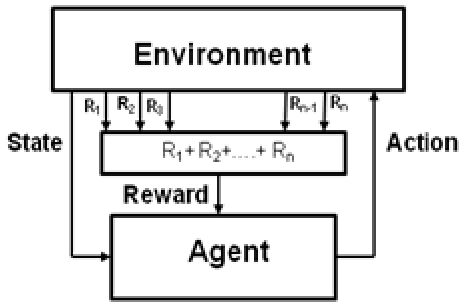

Fig. 5. Hybrid Reward Architecture for RL

III. Results and Discussions

We conduct several experiments in different traffic scenarios and provide a comparative analysis to compare the permanence of different reward functions. A traffic micro-simulator named Simulation of Urban MObility (SUMO) is used with its Python API to create the simulation environment. A four-way road intersection is used where each road has two incoming and two outgoing lanes. The highest speed limit of the incoming lane is 70 km/hour and 40 km/hr for the outgoing lane. The road length is set as 300 meters in the simulation. The vehicles are allowed to pass the intersection into four different routes: (1) from West to East (W-E); (2) from North to South (N-S); (3) from East to West (E-W); and (4) from South to North (S-N).

The DNN is used as a function approximator in the learning process of the simulation. The network structure proposed in [8] is used as a function approximator for all reward functions except [6]. The network structure of [8] is generic enough to fit any number of states and reward functions. For example, if there are *n* numbers of states and *m* incoming lanes, there will be $n \times m$ nodes in the input layer. The number of nodes in the output layer depends on the number of reward functions (x) and many actions (y), which are x×y nodes. The nodes of the hidden layers need to be adjusted according to the number of input and output nodes. The authors have used their network structure to implement their reward function [6]. The parameters used in the learning process are mentioned in Table 1.

Different traffic scenarios are used using synthetic data to compare different reward functions. These traffic scenarios are widely used in the literature [6][8][21]. A total of five configurations are utilized, where the first configuration represents a steady traffic flow with a low traffic rate in all directions. The unstable traffic flow shows in the second configuration where vehicles' arrival rate in the East-West direction is two times less than the North-South direction. The third configuration maintains a steady flow with heavy traffic. The fourth configuration combines the first three configurations to represent the actual live traffic condition with low, heavy, and unstable traffic at different times. The fifth configuration represents the traffic flow for the whole day, considering the traffic variation from 6 AM to 12 AM. Generally, the traffic flow starts with low pressure and increases as the day progresses. It goes to the peak at 9-10 AM and then gradually decreases during the noon. Another peak is created in the evening, which remains longer than the morning peak and gradually decreases during midnight. These five configurations are listed in Table 2.

To get a smooth traffic flow, we want a minimum number of vehicles halting on the road, to wait less time on the road, and to reach the destination in a minimum time. The queue length on the road is also expected to be minimum. As a result, halting number, waiting time, queue length, and travel time are used as parameters to evaluate different rewards. The pseudocode for reward functions comparison can be found as follow.

```
Pseudocode for reward functions comparison
-------------------------------------------
Set $n$ = no. of states, $m$ = no. of lanes, $x$ = no. of reward functions and $y$ = no. of
actions
    Implement DNN with be $n \times m$ nodes in the input layer and $x \times y$ node in the output layer
    Implement the reward halting number $R_{Hn,t}=Hn_{t-1} - Hn_t$
    Implement the reward waiting time $R_{W,t}=W_{t-1} - W_t$
    Implement the reward travel time $R_{T,t}=T_{t-1} - T_t$
    Implement the reward Delay $R_{D,t}=D_{t-1} - D_t$
    Implement HRA using [14]
    Implement Intellilight$R_t = -0.25\sum_{i \in l} L_i - 0.25\sum_{i \in l} D_i - 0.25\sum_{i \in l} W_i - 5C + N + T$
    Implement CRA using [8]
    Implement avg. queue length reward $r_t = |\, max_{i=1,2}\{q_t^{e-w,i}, q_t^{w-e,i}\} - max_{i=1,2}\{q_t^{s-n,i}, q_t^{n-s,i}\}|$
    Implement Metalight [25]
Calculate halting number, waiting time, queue length and travel for all the
implemented reward functions and compare among them
```

We have compared a total of nine reward functions where six of them are single objective, and the rest of the three are multi-objective. Total five traffic scenarios are used to compare the reward functions. The results of the comparisons are depicted from Fig. 6 to Fig. 9. The lower, the better values in the following figures for every performance matrix. The halting number for all configurations is depicted in Figure 6. The result shows that CRA [8] and Intellilight [6] perform competitively to optimize halting numbers. Among the five traffic configurations, CRA performs best

Table 1. Agent's parameters

Parameters	**Values**
Minimum time of the actions	5
Learning rate	0.045
Memory size	1200
Sample size	400
Training epochs	550
Discount factors γ	0.9
$\in$ for exploration	0.05

Table 2. Traffic configurations

Configuration	**Routes**	**Arrival rate (Vehicle/second)**	**Start time (Second)**	**End time (Second)**
1	W-E	0.2	0	7200
	N-S	0.2	0	7200
2	W-E	0.2	0	7200
	N-S	0.4	0	7200
3	W-E	0.4	0	7200
	N-S	0.4	0	7200
4	Configuration 1	0	7200	
	Configuration 2	7201	14401	
	Configuration 3	14401	21600	
5	W-E	0.225	06:00 AM	08:00 AM
	N-S	0.225		
	W-E	0.225	08:00 AM	10:00 AM
	N-S	0.388		
	W-E	0.416	10:00 AM	12:00 PM
	N-S	0.416		
	W-E	0.388	12:00 PM	02:00 PM
	N-S	0.225		
	W-E	0.225	02:00 PM	04:00 PM
	N-S	0.225		
	W-E	0.225	04:00 PM	06:00 PM
	N-S	0.388		
	W-E	0.416	06:00 PM	10:00 PM
	N-S	0.416		
	W-E	0.225	10:00 PM	12:00 PM
	N-S	0.225		

in configurations 1, 2, and 3, and Intellilight [6] performs best in configurations 4 and 5. Halting number, HRA [14], Presslight [11], CRA [8], and waiting time get the second position for the configuration 1, 2, 3, 4, and 5, respectively.

The result for the waiting time comparison for all configurations is shown in Figure 7. The figure shows that CRA [8] performs best in waiting time for all configurations except configuration 5. Presslight [11] wins the first position for configuration 5. There is no single second-best winner for all configurations. For example, HRA [14] performs after CRA [8] in configuration 1, travel time performs second-best result in configuration 2, Intellilight [6], Metalight [25], and CRA [8] is the second position winner for configuration 3, 4, and 5 respectively.

The queue length comparison for all configurations is shown in Figure 8. Like the previous result, there is no single winner for all configurations. Intellilight [6] wins for configurations 4 and 5, CRA [8] shows the best result in configurations 3, and travel time wins for configuration 3. The comparative analysis for travel time is shown in Figure 9. The figure shows that the average travel time for the fifth configuration is higher than the others. The reason is that configuration 5 has a longer time (18 hours) than others. For configuration 1, the HRA [14] performs the best result, although others provide a very competitive result. CRA [8] wins in configurations 2, 3, 4, and 5.

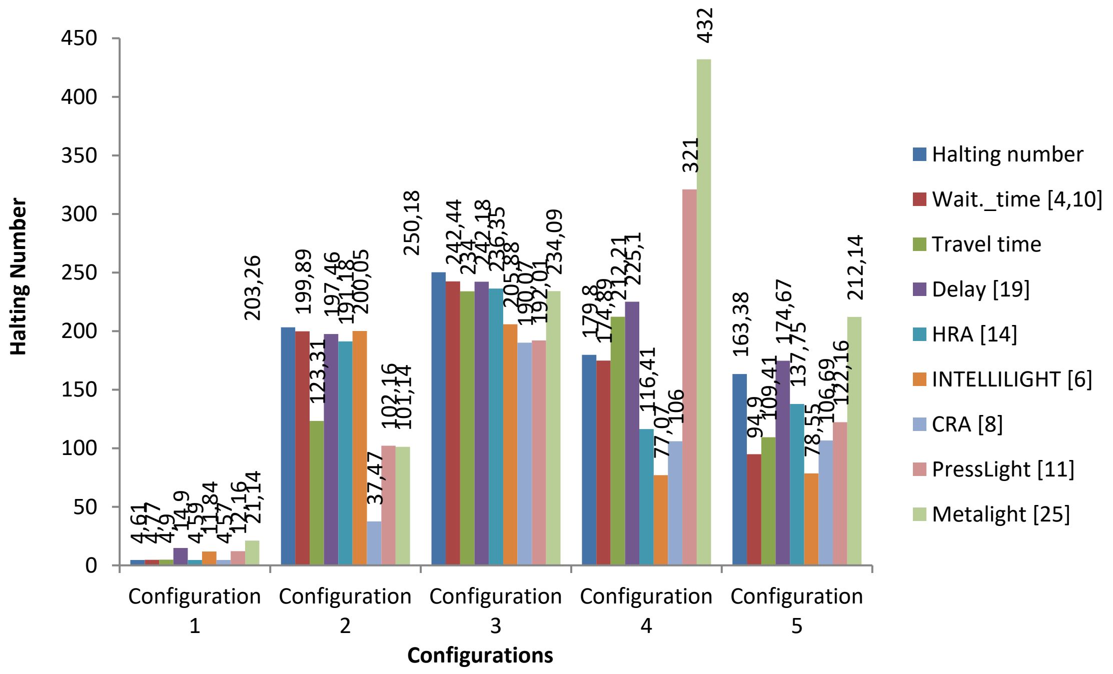

Fig. 6. Comparison of results in terms of average Halting number

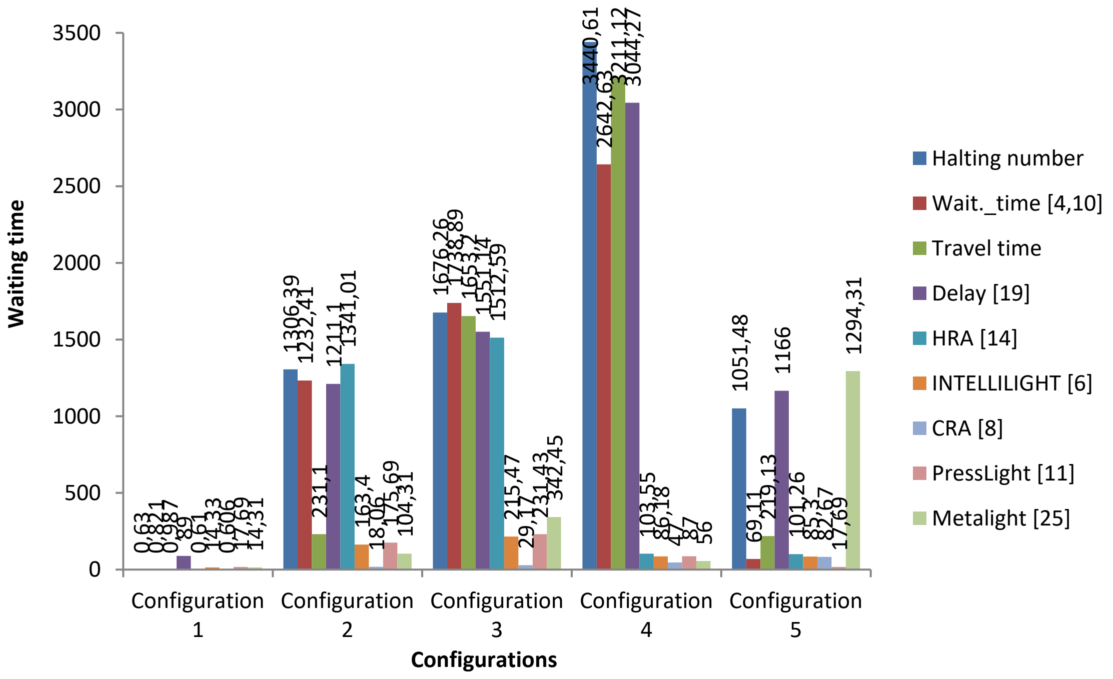

Fig. 7. Comparison of results in terms of average waiting time (s)

Thus, the result concluded that no reward function shows the best result for all parameters. In most cases, either CRA [8] or Intellilight [6] provides the best performance where both of them are multi-objective. In some specific cases, a single objective reward function such as Presslight [11], travel time provides the best result. Thus, from the comparative study, we can conclude that a multi-objective reward function is preferable to a single objective reward function. In addition, among the multi-objective reward functions, CRA [8] or Intellilight [6] perform better among all the configurations.

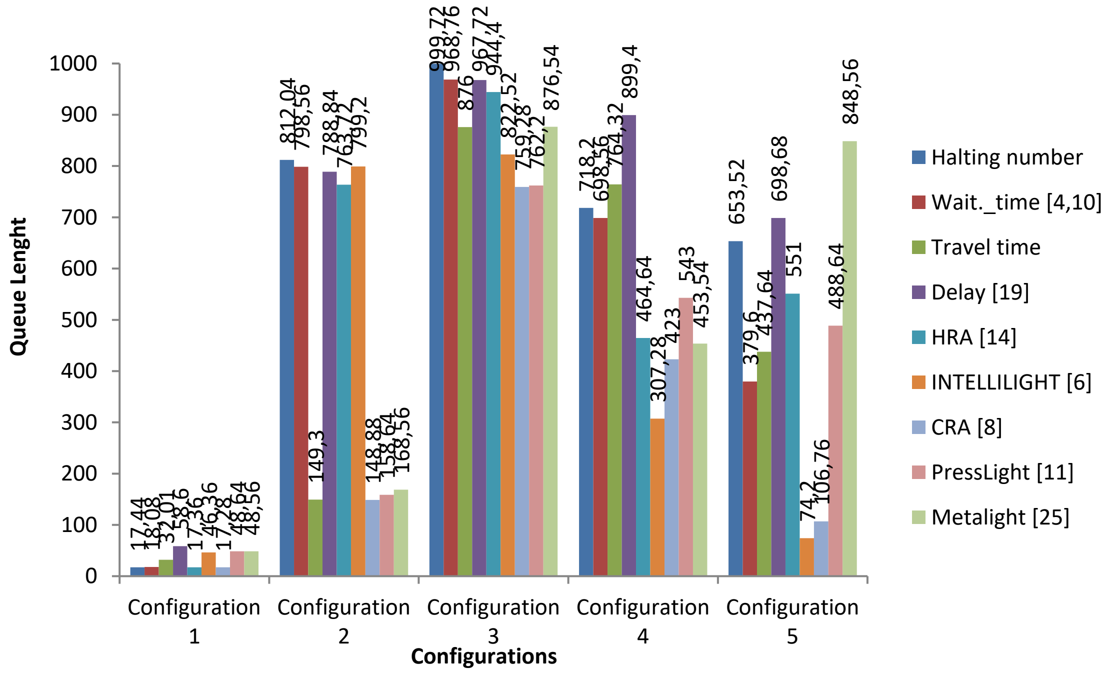

Fig. 8. Comparison of results in terms of average Queue Length

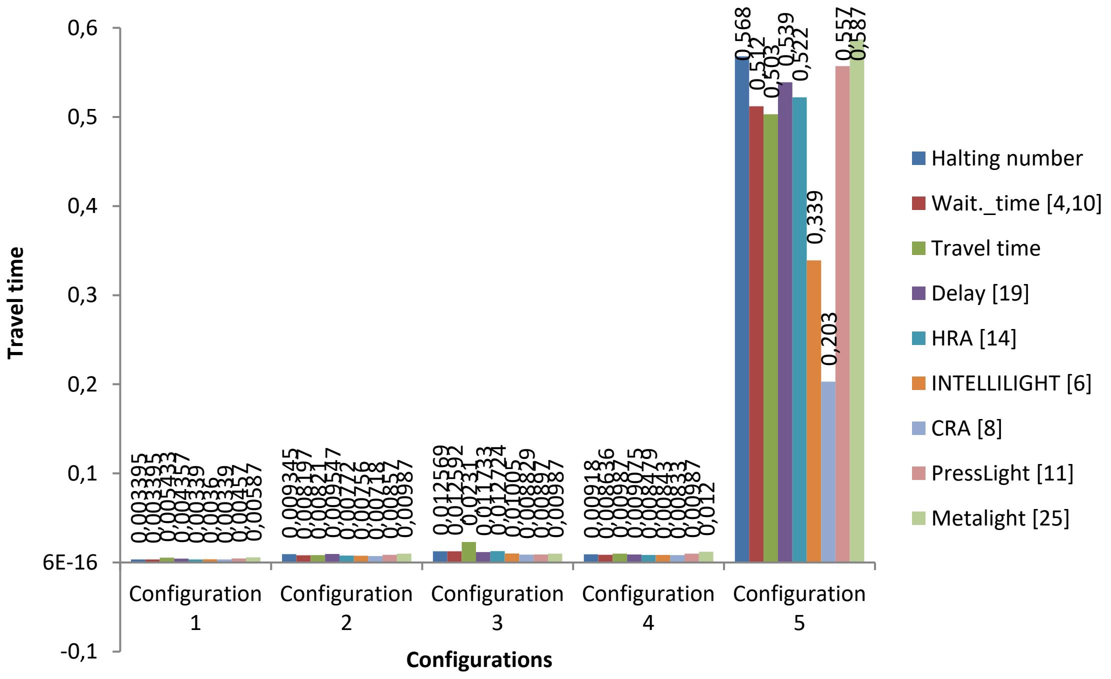

Fig. 9. Comparison of results in terms of average Travel time

IV. Conclusion

The success of DRL highly depends on the reward function since it is used to evaluate the agent's action. If the reward function gives wrong feedback to the agent, the agent will not learn properly. Thus, we need to analyze which reward function performs well in ATSC before using or designing a reward function. In this paper, we have analyzed widely used reward functions and experimentally shown that multi-objective reward function is more suitable for ATSC, and among different multi-objective reward function, CRA and INTELLILIGHT performs well compared with others. In the future, we have a plan to investigate how the network structure impacts designing reward functions.

Declarations

Author contribution

All authors contributed equally as the main contributor of this paper. All authors read and approved the final paper.

Funding statement

This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

Conflict of interest

The authors declare no known conflict of financial interest or personal relationships that could have appeared to influence the work reported in this paper.

Additional information

References

[1] D. Schrank, B. Eisele, T. Lomax, and J. Bak, "2015 Urban Mobility Scorecard," Texas A&M Transportation Institute and inrix, Inc., USA, 2015.

[2] T. Economist, "The Cost of Traffic Jams." https://www.economist.com/ the-economist explains/2014/11/03/the-cost-of-traffic-jams, 2014. Accessed: 2020-02-17.

[3] M. Alam, J. Ferreira, J. and J. Fonseca, "Introduction to Intelligent Transportation Systems", In *Intelligent transportation systems* pp. 1-17 2016, Springer, Cham.

[4] 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.

[5] L. Li, Y. Lv, and F.Y. Wang, "Traffic Signal Timing via Deep Reinforcement Learning". *IEEE/CAA Journal of Automatica Sinica*, 3(3), vol. 3, no. 3, pp. 247-254, Jul 2016.

[6] H. Wei, G. Zheng, H. Yao, and Z. Li, "Intellilight: A Reinforcement Learning Approach for Intelligent Traffic Light Control." *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, Pages 2496–2505, July 2018.

[7] W. Genders and R. Saiedeh, "Using a Deep Reinforcement Learning Agent for Traffic Signal Control," *arXiv preprint arXiv:1611.01142*, 2016.

[8] A. R. M. Jamil, K. K. Ganguly, and N. Nower, "Adaptive Traffic Signal Control System using Composite Reward Architecture Based Deep Reinforcement Learning," *IET Intelligent Transport Systems*, vol. 14, no. 14, pp. 2030–2041, Dec. 2020.

[9] S. Lange and M. Riedmiller, "Deep Auto-encoder Neural Networks in Reinforcement Learning," *The 2010 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8, July 2010.

[10] X. Liang, X. Du, G. Wang, and Z. Han, "Deep Reinforcement Learning for Traffic Light Control in Vehicular Networks," arXiv preprint arXiv:1803.11115, 2018.

[11] C. Chen *et al.* "Toward a Thousand Lights: Decentralized Deep Reinforcement Learning for Large-scale Traffic Signal Control", *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 34, No. 04, pp. 3414-3421, Apr. 2020.

[12] A.R.M. Jamil, K.K Ganguly, N. Nower, "An Experimental Analysis of Reward Functions for Adaptive Traffic Signal Control System", *Proceedings of the International Conference on Distributed Sensing and Intelligent System (ICDSIS)*, Springer, 2020.

[13] P. Mannion, J. Duggan, and E. Howley, "An Experimental Review of Reinforcement Learning Algorithms for Adaptive Traffic Signal Control", *Autonomic Road Transport Support Systems. Springer*, pp. 47–66, 2016.

[14] H. Van Seijen *et al.*, "Hybrid Reward Architecture for Reinforcement Learning" *Proceedings of the Advances in Neural Information Processing Systems*, pp. 5392–5402, 2017.

[15] S. B. Kotsiantis, I. Zaharakis, and P. Pintelas, "Supervised Machine Learning: A Review of Classification Techniques," *Proceeding of the 2007 Conference Emerging artificial intelligence applications in computer engineering*, vol. 160, pp. 3–24, 2007.

[16] H. B. Barlow, "Unsupervised Learning," *Neural computation*, vol. 1, no. 3, pp. 295–311, 1989.

[17] S. Lange and M. Riedmiller, "Deep Auto-encoder Neural Networks in Reinforcement Learning," *in The 2010 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8, IEEE, 2010.

[18] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized Experience Replay," arXiv preprint arXiv:1511.05952, 2015.

[19] E. Van der Pol and F. A. Oliehoek, "Coordinated Deep Reinforcement Learners for Traffic Light Control," *Proceedings of Learning, Inference and Control of Multi-Agent Systems (NIPS)*, 2016.

[20] J. van Dijk, "Recurrent Neural Networks for Reinforcement Learning: An Investigation of Relevant Design Choices," 2017.

[21] M. Coskun, A. Baggag, and S. Chawla, “Deep Reinforcement Learning for Traffic Light Optimization,” in *2018 IEEE International Conference on Data Mining Workshops (ICDMW)*, pp. 564–571, IEEE, 2018.

[22] M. A. Khamis, W. Gomaa, and H. El-Shishiny, “Multi-objective Traffic Light Control System Based on Bayesian Probability Interpretation,” in *2012 15th International IEEE Conference on Intelligent Transportation Systems*, pp. 995–1000, IEEE, 2012.

[23] M. A. Khamis and W. Gomaa, “Adaptive Multi-objective Reinforcement Learning with Hybrid Exploration for Traffic Signal Control Based on Cooperative Multi-agent Framework,” *Engineering Applications of Artificial Intelligence*, vol. 29, pp. 134–151, 2014.

[24] A. Vidali, L. Crociani, G. Vizzari, and S. Bandini, “A Deep Reinforcement Learning Approach to Adaptive Traffic Lights Management,” in *Proceedings of the 20th Workshop” From Objects to Agents”*, Parma, Italy, 2019.

[25] X. Zang, H. Yao, G. Zheng, N. Xu, K. Xu, and Z. Li, "Metalight: Value-based Meta-reinforcement Learning for Traffic Signal Control," *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. No. 01, pp. 1153-1160, Apr. 2020.