Int. j. inf. tecnol. (April 2024) 16(4):2273–2282
https://doi.org/10.1007/s41870-023-01545-8

ORIGINAL RESEARCH

Mitigating congestion in multi-agent traffic signal control: an efficient self-attention proximal policy optimization approach

Oussama Chergui[1,2] · **Lamri Sayad**[3]

Received: 13 May 2023 / Accepted: 16 September 2023 / Published online: 8 November 2023

Abstract Traffic congestion is a persistent problem that effects cities worldwide, necessitating innovative solutions. This paper presents a novel traffic light control system using multi-agent proximal policy optimization with self-attention. Our approach outperforms traditional methods by 30% in reducing waiting times in high-traffic demand scenarios. By utilizing transfer learning and encoding mechanisms for dynamic input size adaptation, our approach enables scalability to larger networks without the need for costly training. This study underscores the potential of our approach as a dependable solution for addressing large-scale traffic congestion challenges.

Keywords Traffic lights control · Multi-agent · Deep reinforcement learning · Proximal policy optimisation · Self attention · Transfer learning

1 Introduction

The last century has seen large improvements in healthcare and the living conditions of the average person, leading to a boom in population growth, which the majority of are located in cities. This trend is expected to continue in the coming years [1]. due to this ever-increasing population growth and the influx of vehicles every year, cities are faced with a range of financial and environmental issues caused by traffic congestion. Economically, congestions caused a total of 4.3 billion hours of delay and 1.7 gallons of fuel waste which is equal to $101 billion in loss [2] and 100 billion euros a year losses in Europe [3]. Environmentally, According to the United States Environmental Protection Agency, transportation is the largest contributor of greenhouse gas emissions in the US at 27 percent in 2020, with vehicles such as cars and light-duty trucks accounting for majority of those emissions. Aside from the environmental and financial losses incurred, congestion has been found to have a detrimental effect on drivers' satisfaction levels and mental well-being [4] with 95% of the drivers negatively affected by traffic congestion with a score of the stress level at 3.23 ± 0.71 [5].

Intelligent Transportation Systems (ITS) stand as a promising solution to the pressing issue of traffic congestion, drawing extensive research attention in recent decades. These approaches have substantiated their effectiveness, yielding annual savings of $4.7 billion in US cities [6]. They achieve this by improving drivers' decision-making through smart vehicle routing [7], reducing accident rates [8, 9] and optimizing smart parking systems [10]. Furthermore, the implementation of dynamic traffic light scheduling, referred to as Adaptive Traffic Light Control (ATLC), plays a pivotal role. These systems utilise the latest sensor and wireless communication technologies, such as Vehicle-to-Vehicle (V2V) and Vehicle-to-Infrastructure (V2I) systems [11], to fine-tune real-time traffic light signals based on data acquired from vehicles, roadside units (RSUs), or passengers. This data reflects aspects such as traffic demand, vehicle speed, waiting time, and lengths of incoming lane

✉ Oussama Chergui
oussama.chergui@univ-msila.dz

[1] Laboratory of Signals and Systems Analysis, Faculty of Technology, University of M′sila, PO Box 166 Ichebilia, 28000 M′sila, Algeria

[2] Faculty of Mathematics and Computer Science, University of M′sila, PO Box 166 Ichebilia, 28000 M′sila, Algeria

[3] Laboratory of Informatics and its Applications of M′sila (LIAM), Faculty of Mathematics and Computer Science, University of M′sila, PO Box 166 Ichebilia, 28000 M′sila, Algeria

queues which are then processed by algorithms that generate optimal plans for intersections with the aim of reducing city-wide congestion.

However, the algorithms used in ATLC face the challenge associated with maintaining their superior performance as traffic volumes rise. In particular, getting good results in a low traffic volume will not guarantee keeping these results in a higher traffic volume representing rush hours. Other challenges that impede their effectiveness are scalability and portability issues. In a multi-agent setting, where each agent represents an intersection, an algorithm that was trained to fit a smaller network may not deliver the desired outcome on the larger network. Reinforcement learning (RL) algorithms have emerged as a potential solution to reduce traffic congestion and waiting times by controlling traffic signals. Different types of RL algorithms, such as Q-learning, Deep Reinforcement Learning, end Proximal Policy Optimisation, can improve performance without requiring agents to learn by engaging in laborious data collection campaigns. While these algorithms show great promise in reducing waiting time, there is still room for improvement.

In our work, we aim to address those issues by first applying our Self-Attention Multi-Agent Proximal Policy Optimization (SA-MAPPO) algorithm to a six- intersections network with normal and high traffic volumes. Subsequently, we will repeat the same process with a twelve-intersection network with minimal modifications to the algorithm and lower costs and time compared to the cost and time spent training a smaller network using the previous pre-trained algorithm. The contributions of this work are:

- Using attention mechanism alongside Proximal Policy Optimization (PPO) to extract pertinent features from raw data, and compare it with the baseline fixed algorithm and PPO without attention
- Prove the scalability of the algorithm by testing it in higher traffic flow and demonstrating increased efficacy in larger networks while incurring minimal training costs via a pre-trained PPO combined with a small encoder network to adjust the the dymanic input.

2 Related works

Research in traffic control using reinforcement learning has undergone extensive exploration, encompassing a variety of approaches and applications.

In certain approaches, algorithms are tailored to one specific networks. For instance, Wiering et al. [12] focused on optimizing traffic lights by predicting estimated waiting times for cars at 15 intersections network. Their work achieved a notable 25% enhancement in a Green Light District (GLD) simulator without altering network size or traffic flow. Instead of employing a neural network, they utilized a car-based value function.

Arel et al. [13] applied an RL method using a Longest Queue First (LQF) algorithm executed by local agents to minimize delay and cross-blocking likelihood in a 5 intersections environment. Using the AIMSUN simulation tool in [14], a Q-learning-based RL algorithm was employed to investigate a non-grid 50-intersection network. While it demonstrated impressive results in high-flow traffic, its effectiveness in regular traffic demand was limited.

Exploring an alternative methodology for state representation, SS Mousavi [15] utilized image snapshots of traffic situations in conjunction with deep learning RL within the SUMO simulation. While this approach demonstrated promising results and ease of implementation, it posed challenges in terms of multi-agent applicability and scalability.

Calvo et al. [16] adopted the Independent Deep Q-Network (IDQN) to control traffic lights in a heterogeneous network with three intersections featuring varying traffic demands. In low traffic conditions, the algorithm outperformed baseline fixed approaches, although adjustments were needed for high-demand scenarios.

Other researchers applied their algorithms to diverse networks through retraining. For instance, Chu et al. [17] introduced a decentralized multi-agent reinforcement learning algorithm based on Actor Critic (A2C). They initially trained their model on a 25-intersection synthetic network and on a 30-intersection network. This approach utilized policies from neighboring agents to improve observability and achieved success in both network scenarios. Nevertheless, it was observed that retraining for larger networks incurred substantial costs.

In a large-scale implementation [18], Deep Q network RL was trained in a 4 grid network and a Manhattan road network lights using the City-Flow simulation tool. It showcased an impressive 80% improvement in average delay over fixed method.

Yang et al. [19] introduced a modified version of the Multiagent Proximal Policy Optimization Algorithm to control traffic lights and regulate vehicle speed across various scenarios and trained their algorithm on a 6 and 9 intersections, outperforming regular PPO in reward-based metrics.

Wang et al. [20] aimed to capture temporal states using recurrent neural networks and spatial states with graph networks integrated with RL. Their innovative approach achieved a remarkable 200% improvement in regular traffic flow over fixed approaches.

Li et al. [21] utilized a network-sharing mechanism in their Deep Deterministic Policy Gradient reinforcement learning algorithm for network-wide adaptive traffic control. This approach resulted in a 100% improvement in queue

length for a 10×10 grid network and a 30% improvement for a Montgomery County real data network compared to fixed-time traffic control.

In [22], the authors introduced the 'Friend-Deep Q Network,' a multi-agent cooperative approach for traffic signal control in 2–4 junctions scenarios using SUMO simulations. It outperformed both independent Q-networks and fixed networks.

Lastly, some studies leveraged the power of transfer learning by initially training their algorithms on a specific network and applying pre-trained weights to other networks while maintaining a consistent input size. A cooperative group-based RL algorithm in [23] focused on large-scale scenarios, utilizing k-neighbors based joint state representation and distance factors for reward formulation. Initially trained in a multi-agent setting on a grid network, this approach achieved a 100% improvement during peak hours on grid networks compared to fixed approaches.

In another approach, as demonstrated by Mo et al. in [24], training commenced with a single agent and subsequently extended to encompass multiple agents within a grid network. Scalability was enhanced through the incorporation of pre-trained models into the Asymmetric Advantage Actor-Critic algorithm. These models, based on data from connected vehicles, underwent rigorous testing across various traffic scenarios, including 5×5 and 2×2 grid networks.

In Our research, we used transfer learning by initially training our multi agent algorithm on a 6-intersection network and then applied its pre-trained weights to a 12-intersection network. What distinguishes our approach is the incorporation of an encoder network, which dynamically adjusts input dimensions, departing from the conventional practice of maintaining uniform input sizes. enhancing our algorithm's adaptability and scalability.

We summarize the details of Reinforcement learning based implementation in Table 1

3 Proposed method

In this section, we will define the algorithms underlying our approach and then provide a detailed explanation of our implementation.

3.1 Background

Reinforcement learning: (RL) is a machine learning algorithm, alongside supervised and unsupervised learning, primarily utilized for solving combinatorial optimization and robotics-related problems.

In the context of RL, the agent, or a set of agents, interacts with an environment by taking actions at time step based on the received state and subsequently receives rewards or penalties, enabling it to learn an optimal policy that maximizes the total expected rewards [25].

Policy gradients: are a class of reinforcement learning algorithms that leverage neural networks for policy approximation and utilize gradient ascent to maximize rewards and optimize the network. Actor-critic methods, a subset of policy gradients, simultaneously learn value functions to help finding the best policy [26]:

- **Actor**: refer to the neural network responsible for updating the policy $\pi_\theta(a \vee s)$. It optimizes parameters to select the best action a given a particular state s.
- **Critic**: responsible for learning the value function $V(S)$ or state-value function $Q(a, s)$.

Proximal policy optimization(PPO): PPO is a policy gradient algorithm that aims to fix the instability in training that is caused by sampling inefficiency and large gradient steps in policy gradient algorithms [27, 28]. The objective function of PPO is given by Eq. (1):

$$L^{CLIP}(\theta) = E_t\left[min\left(r_t(\theta)A_t, clip\left(r_t(\theta), 1-\in, 1+\in\right)A_t\right)\right] \quad (1)$$

$$r(\theta) = \frac{\pi_\theta(s)}{\pi_{\theta_{old}}(s)} \quad (2)$$

$$A(s,a) = Q(s,a) - V(s) \quad (3)$$

r represents the ratio between the old policy and the new policy, measuring the difference between them as shown in Eq. (2). A is the advantage function Eq. (3) which is used to reduce the variance of the estimation of the value function. The objective function is obtained by multiplying the ratio with the advantage.

To discourage large updates in case the ratio is too large, The *min* function is used in the objective function in Eq. (1) to choose between the objective function or the clipped version of the objective function. This ensures the algorithm's stability during convergence.

Self-Attention: self- Attention is a transformer-based neural network used primarily in language models [29] that can be applied across multiple fields. self-Attention takes a matrix $x = \left(x_1, \ldots x_n\right)$ as an input and performs calculations

Table 1 Summarization of related works

References	Year	Method	Network size and traffic flow	Simulation tool	Reward
[12]	2004	Multi agent Reinforcement learning	15 INTERSECTIONS no high traffic flow	Green Light District (GLD)	Minimize waiting time by assigning values based on movement and delays of cars
[13]	2010	longest-queue-first (LQF) algorithm for actions, Q-leaning for value function	5 intersections dynnamic traffic flow	Matlab based simulator	Agent is rewarded for delay reduction, penalized for higher average delay
[14]	2010	Q-Learning	50 intersections dynamic traffic flow	Aimsum	Inverse of approaching queue length
[15]	2017	Deep policy-gradient and value-function based agents	1 intersection, regular traffic flow	SUMO	Difference in cumulative delays between consecutive actions
[16]	2018	Independent Deep Q-Network	3 intersections, dynamic traffic flow	SUMO	Inverse cumulative waiting time, range in [0,1]
[17]	2019	actor-critic based multi agent RL	25 intersection synthetic grid as well as 30 intersection based on real world city monaco dynamic traffic flow	SUMO	Negative of the sum of Queue and waiting time
[18]	2020	Deep Q network with parameters sharing	a 4×4 grid network and Manhatten road network dynamic traffic flow	City-Flow	Pressure: difference between The sum of queued vehicles at exit and entry lanes
[19]	2021	Multi agent Proximal Policy Optimization	6 and 9 intersections Regular traffic flow	Flow	Negative value of average delay
[20]	2022	RL algorithm integrated with Recurrent neural network and graph network	6×6 grid network and real word networks dynamic traffic flow	City-Flow	Negative value of queue length
[23]	2021	Cooperative group-based Q learning algorithm	6×6 grid network Monaco and Harbin synthetic road network, dynamic traffic flow	SUMO	Assigning values [− 1.0.1] based on average flow of vehicles
[21]	2021	Deep Deterministic Policy Gradient with network sharing mechanism	10×10 grid network and a real-world network based on downtown Montgomery County, Maryland Dynamic traffic flow	SUMO	Difference in average delays between consecutive actions
[24]	2022	Asymmetric Advantage Actor-critic and pretrained algorithm	5×5 and 2×2 grid network Dynamic traffic flow	SUMO	Negative of the difference between the sum of normalized queued vehicles at entry and exit lanes
[22]	2023	Deep Q network	2 and 4 intersections Dynamic traffic flow	SUMO	Difference of the mean delay of vehicles between states

on it to focus attention on certain input vectors over others, allowing for complex patterns and dependencies within the data to be identified. Through its calculations, it returns a matrix of the same size $y = (y_1, \dots y_n)$ as an output which can then be used to inform decision-making processes.

The process involves calculating the key K, queries Q, and values V by multiplying the input X with the respective weights W [30] as given by in the equations Eqs. (4–6):

$$k_i = X_i W^k \tag{4}$$

$$Q_j = X_j W^q \tag{5}$$

$$V_j = X_j W^v \tag{6}$$

Next, in Eq. (7) we perform scaled dot products with the keys and the queries. Scaling the dot product speed up the computation and save memory space [31]:

$$e_{ij} = \frac{K_i Q_j^T}{\sqrt{d_k}} \tag{7}$$

Finally, the *softmax* function is applied to obtain attention scores in Eq. (7), which are then combined with the values to yield the final output.

$$y_i = \sum_{j=1}^{n} softmax(e_{ij}) V_j \tag{8}$$

Encoder: Following the self-attention layer, we employed a Convolutional Neural Network (CNN) layer as an encoder. CNN is a neural network which performs dot product operations between kernels or filters and tensor inputs to create a new tensor which typically contains newly generated features. In this work, we utilized the CNN to compress the input layer into a size that adequately fits the Pre-trained MAPPO without sacrificing too much information.

3.2 Representation

The representation of state, actions, and rewards in a reinforcement learning (RL) algorithm is crucial for success, enabling the modeling of the environment and appropriate incentivization of behavior. Our representation is as follows:

State: The input of the attention network is a matrix $K(N \times M) = (S_0 \dots S_N)$ where N is the number of intersections (or agents) in the simulation and M is the number of features of each agent. Each vector is the local state S and is represented by:

$$S_i = (currentphaseId, duration, Hi, X_i) \tag{9}$$

The first element of the vector in Eq. (9) is the Id of the current phase which is the combination of green, red, and yellow lights that dictate whether vehicles are allowed to move at the current time. duration is the current phase duration in steps. H is the number of halting vehicles in every lane connected to intersection i. $X_i = (x_{0i} \dots x_{ii} \dots x_{ni})$ is a vector denoting neighborhood connection where each element signifies the presence $(x_{ij} = 1)$ or absence $(x_{ij} = 0)$ of edge connection between intersections i and j.

After passing through the attention network, each vector S_i of the matrix K is passed to the policy network separately. as for the value function input, matrix K is transformed into a flattened vector, to which we append another vector $A = (a_0 \dots a_N)$ that signifies the current actions of all agents.

Actions: Each agent in the network has two possible actions:

- 1: Extend the current phase by 4 steps if it is a green phase.
- 0: jump to the next phase.

A constraint was imposed where Each phase cannot have a duration shorter than 4 s or longer than 60 s.

Rewards: We use a global reward function that represents the whole network

$$R = (W \times J) \tag{10}$$

W is the average accumulated waiting time of all vehicles in the simulation, and J is the sum of average vehicles length across all lanes in the previous step. These metrics provide valuable insights into the traffic from the simulation's beginning to the current time step. W captures all waiting time for all vehicles since entering the simulation, whereas J offers a more current perspective. Combining these metrics optimally informs our traffic control algorithm's decision-making for next actions.

The pseudocode in Algorithm 1 describes in details the flow and inner working of our method. This particular implementation is applied on the initial network upon which the algorithm is initially trained. When transitioning to a larger network, we adopt a strategy where we freeze all the layers of the MAPPO model and use their pre-trained weights for the larger network.

To adapt to the increased input size of this larger network, we introduce an additional encoder layer, implemented as a Convolutional Neural Network. This encoder

serves the purpose of reducing the input dimensions to align with the original smaller MAPPO network, ensuring compatibility without sacrificing critical information. This approach allows us to effectively leverage the knowledge acquired from the initial network and apply it to more complex network.

Algorithm 1 SA-MAPPO

Initialization:
Initialize the Self Attention network parameters SA, policy network θ and value network φ.
max episodes M
max steps MS
memory buffer
for each episode 1 ... M **do**
  for each step t….MS **do**
    Collect the states o_t and get the policy network input o'_t from the self-attention network Eq.(8)
    for each agent i in environment **do**
      Get next action of this agent a_{ti} from the policy network $\pi(a_{ti}|o'_{ti})$
    end for
    Collect the rewards r_t Eq. (10) and next states o_{t+1}based on actions a_t.
    Get $V(o_t,a_t)$ from the critic network
    Calculate Advantage estimate Eq. (3)
    Add (o_t, a_t, o_{t+1}, r_t) to the memory buffer
  end for
  Calculate the objective function from Eq. (1) and Update the SAMAPPO algorithm using Adam Optimizer
  Clear the memory buffer
end for

4 Simulation and training

4.1 Environment setting

In this work, we use the SUMO-RL library, developed on top of 'Simulation of Urban MObility' (SUMO), a well-established traffic simulation tool. This selection is made to facilitate seamless integration and compatibility with popular reinforcement learning algorithms [32].

we employ two networks: a six-intersection and a 12-intersection network, depicted in Fig. 1 Each intersection connects to four edges and has eight incoming lanes.

We use two types of vehicles as expressed in Table 2.

Lastly, and to simulate rush hours we used two traffic scenarios to test our algorithm, the default scenario here we have 1 vehicle every 1.2 timesteps, and the higher flow where we have 1 vehicle per 0.8 timesteps.

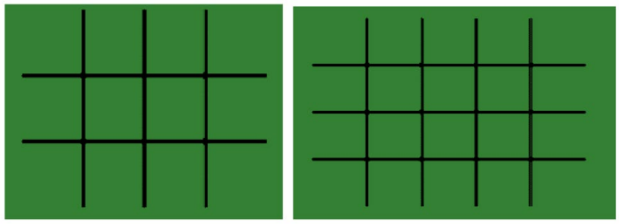

Fig. 1 Maps used in the simulation. The left is the 6 intersections and the right one is 12 intersections

4.2 Evaluation metrics

In our evaluation, we assess the proposed algorithm using four metrics:

Acc. Waiting Time: This metric aggregates the accumulated waiting time of vehicles in all lanes, recorded at each time step.
Avg. Waiting: Calculated as the average waiting time of all vehicles throughout the simulation collected every step.
Avg Speed: This metric represents the mean speed of all vehicles during the simulation, divided by the number of steps.

Each simulation has a fixed duration of 3600 s, equivalent to 900 steps since every step is equal to 4 s. To benchmark our algorithm (SAMAPPO), we compare its performance against the following algorithms:

- **MAPPO:** A multi-agent PPO without self-attention layers.
- **PRESSURE:** SAMAPPO, with pressure-based reward functions, following the approach introduced in [24, 33].
- **FIXED:** This serves as the baseline algorithm, implementing fixed timing for traffic lights.

4.3 Training

In our analysis of training results, since the models under evaluation do not share the same reward function. We gauge their performance by measuring waiting times during training, as depicted in the training curve plots in Figs. 2 and 3.

Table 2 Vehicles proprieties in the simulation

Type	Max speed	Length	Probability
Car	35	4.5	0.9
Bus	25	12.6	0.1

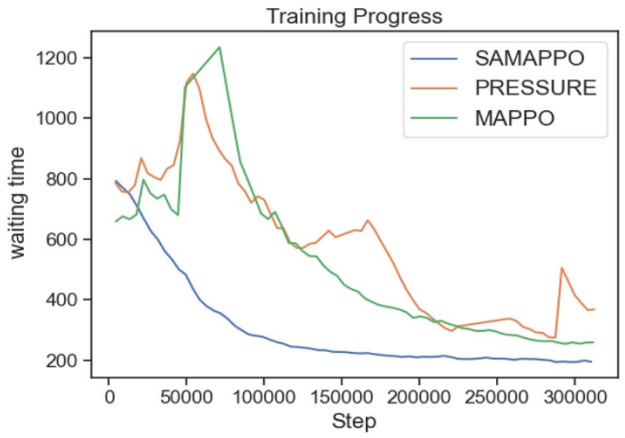

Fig. 2 The mean waiting time during training the algorithm on 6 intersections

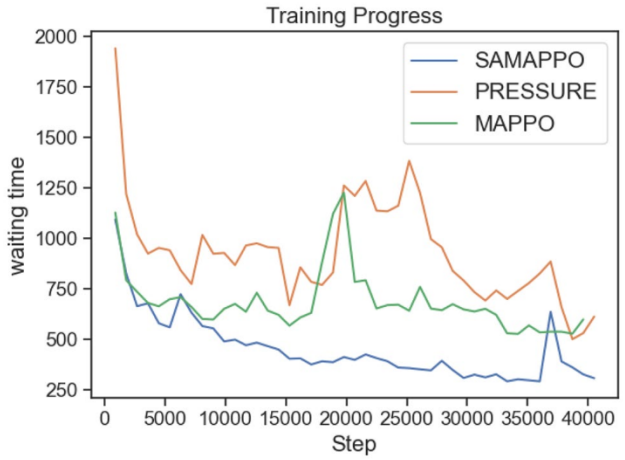

Fig. 3 The mean waiting time during training the algorithms on 12 intersections

Our observations reveal that all models exhibit a consistent reduction in waiting times as training progresses. Notably, the SAMAPPO algorithm outperforms other algorithms, consistently achieving lower waiting times and demonstrating more stable training across both the 6-intersection and 12-intersection networks.

A noteworthy finding is the faster convergence to the minima in the waiting time curves for the 12-intersection network, taking less than 8 times compared to the 6-intersection network. This fast convergence is attributed to our transfer learning approach, leveraging pretrained models from the 6-intersection network to help feature construction in the 12-intersection scenario, with training primarily focused on the encoder layers while the PPO network layers are frozen.

Table 3 Results on 6 intersection and normal traffic flow

	Acc. waiting	Avg speed	Avg waiting
SAMAPPO	**19,743**	**9.18**	**201**
PRESSURE	26,466	8.95	327
MAPPO	21,920.3	9.11	249
FIXED	181,460	6.4	2614

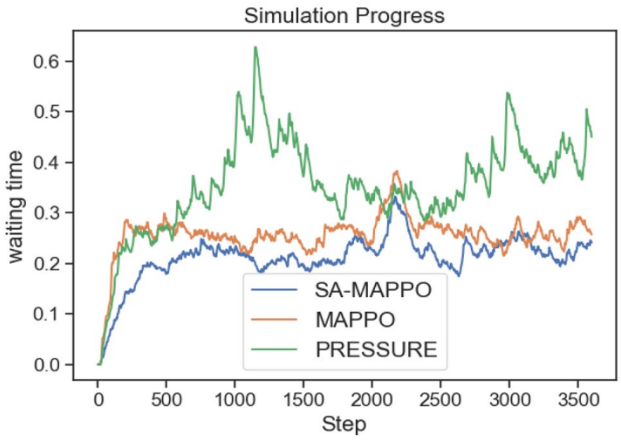

Fig. 4 Waiting time for all algorithms in 6 intersection and regular traffic flow during the simulation

Table 4 Results on 6 intersection and high traffic flow

	Acc. waiting	Avg speed	Avg waiting
SAMAPPO	**42,420**	**8.34**	**314**
PRESSURE	50,069	8.07	465
MAPPO	49,674	8.20	451
FIXED	233,840	6.13	4239

5 Results and discussion

Throughout the results tables, the top results are highlighted in bold for clarity. In the context of 6-intersection networks, our SAMAPPO algorithm consistently demonstrates superiority across benchmark metrics as evident in Table 3 and Fig. 4, achieving approximately a 19% improvement in mean waiting time over its nearest competitor, MAPPO. Notably, our choice of reward function outperforms the Pressure reward function by 38% under regular traffic flow conditions.

From Table 4 we see that in high traffic flow scenarios, algorithms incorporating self-attention layers exhibit greater resilience compared to non-self-attention-based MAPPO. SAMAPPO, for instance, experiences a 36% performance decrease in higher flow conditions, whereas MAPPO's

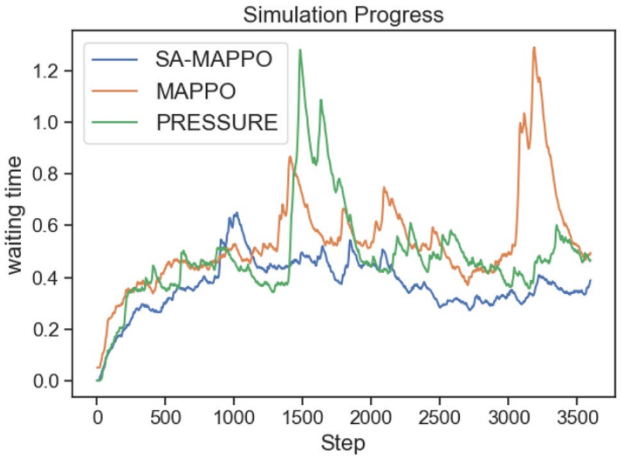

Fig. 5 Line plots of system waiting time for all algorithms in 6 intersection and high traffic flow

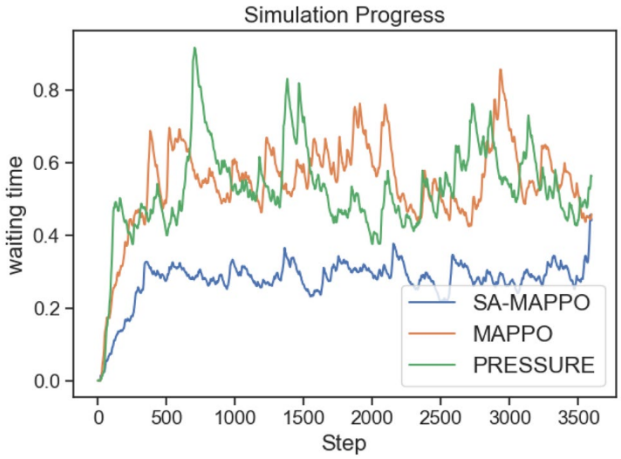

Fig. 6 Line plots of system waiting time for all algorithms in 12 intersection and regular traffic flow

Table 5 Results on 12 intersection and regular traffic flow

	Acc. waiting	Avg speed	Avg waiting
SAMAPPO	**33,411**	**9.39**	**263**
PRESSURE	46,707	9.07	493
MAPPO	47,836	9.09	505
FIXED	228,844	7.59	3590

Table 6 Results on 12 intersection and high traffic flow

	Acc. waiting	Avg speed	Avg waiting
SAMAPPO	**59,192**	**8.98**	**376**
PRESSURE	70,886	8.79	573
MAPPO	80,884	8.6	657
FIXED	316,929	7.25	4761

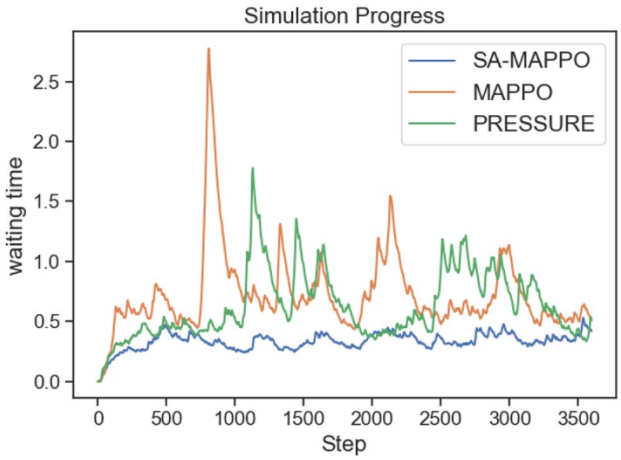

Fig. 7 Line plots of system waiting time for all algorithms in 12 intersection and high traffic flow

performance deteriorates by almost 45%, as evident in Fig. 5, where SAMAPPO's curve displays greater stability compared to MAPPO's.

This resilience against higher traffic flow was also observed in the PRESSURE algorithm which employs self-attention layers, where the performance dropped by only 30%, reinforcing the efficacy of self-attention mechanisms in managing high traffic flow situations.

In the context of the 12-intersection network, we observe from Tables 5 and 6 that our trained algorithms maintain their performance with the inclusion of the transfer learning and encoding mechanism, as they consistently outperform the fixed approach by similar margin as with 6 intersections. Specifically, we notice from as Figs. 6 and 7 that our approach continues to demonstrate superior performance over other algorithms, surpassing Pressure and MAPPO by 46% and 47%, respectively, in normal traffic flow scenarios, and by 34% and 42% in high traffic flow scenarios, as measured by the average waiting time.

It is worth highlighting that the PRESSURE algorithm exhibits superior waiting time performance compared to MAPPO across both traffic flow scenarios, compared to 6 network results. This difference can be attributed to the incorporation of self-attention layers in PRESSURE, facilitating the extraction of essential features from multiple agents, particularly advantageous in larger networks.

6 Conclusion

In this paper, we introduced a self-attention-based Proximal Policy Optimization (PPO) algorithm for tackling traffic congestion in a multi-agent context. Our research not only outperforms traditional methods but also demonstrates scalability to handle increased traffic and larger networks with minimal modifications and training time. Even when the input network shape changes, offering both superior performance and adaptability to meet the demands and complexity of urban environments.

6.1 Future scope

In this study, we conducted experiments on homogenous intersections, each featuring an identical configuration of incoming and outgoing lanes. Looking ahead, there exist several promising avenues for future research. Firstly, extending our methodology to heterogeneous networks, where intersections vary in shape and complexity, can provide valuable insights into its adaptability across diverse urban landscapes. Moreover, transitioning from controlled settings to real-world networks, which may comprise more than twelve intersections, will enhance the applicability of our approach.

Funding The authors declare that no funding was received to assist with the preparation of this manuscript.

Data availability The dataset used in this work was generated using SUMO and can be found in this link: https://github.com/cherouss/SAMAPPO/data

Code availability The code of this work can be found in this repository: https://github.com/cherouss/SAMAPPO.

References

1. Albino V, Berardi U, Dangelico RM (2015) Smart cities: definitions, dimensions, performance, and initiatives. J Urban Technol 22:3–21
2. Schrank D, Albert L, Eisele B, Lomax T (2021) Urban Mobility Report. Texas A&M Transportation Institute, College Station
3. Christidis P, Rivas NI (2012) Measuring road congestion. Institute for Prospective Technological Studies, European Commission Joint Research Centre
4. Higgins CD, Sweet MN, Kanaroglou PS (2018) All minutes are not equal: travel time and the effects of congestion on commute satisfaction in Canadian cities. Transportation 45:1249–1268. https://doi.org/10.1007/s11116-017-9766-2
5. Fattah MdA, Morshed SR, Kafy A-A (2022) Insights into the socio-economic impacts of traffic congestion in the port and industrial areas of Chittagong city. Bangladesh Transport Eng 9:100122. https://doi.org/10.1016/j.treng.2022.100122
6. ChengAaron Z, Pang M-S, Pavlou PA (2020) Mitigating traffic congestion: the role of intelligent transportation systems. Inf Syst Res 31:653–674. https://doi.org/10.1287/isre.2019.0894
7. Agarwal V, Sharma S (2023) DQN Algorithm for network resource management in vehicular communication network. Int J Inf Technol. https://doi.org/10.1007/s41870-023-01399-0
8. Yadav R, Dahiya PK, Mishra R (2020) Comparative analysis of automotive radar sensor for collision detection and warning system. Int J Inf Technol 12:289–294. https://doi.org/10.1007/s41870-018-0167-3
9. Mohapatra H, Rath AK, Panda N (2022) IoT infrastructure for the accident avoidance: an approach of smart transportation. Int J Inf Technol 14:761–768. https://doi.org/10.1007/s41870-022-00872-6
10. Sharma R, Singh U (2021) Fuzzy based energy efficient clustering for designing WSN-based smart parking systems. Int J Inf Technol 13:2381–2387. https://doi.org/10.1007/s41870-021-00789-6
11. Wang Y, Yang X, Liang H, Liu Y (2018) A review of the self-adaptive traffic signal control system based on future traffic environment. J Adv Transp 2018:e1096123. https://doi.org/10.1155/2018/1096123
12. Wiering MA, Veenen J van, Vreeken J, Koopman A (2004) Intelligent traffic light control. In: Utrecht University: Information and Computing Science
13. Arel I, Liu C, Urbanik T, Kohls AG (2010) Reinforcement learning-based multi-agent system for network traffic signal control. IET Intell Transp Syst 4:128. https://doi.org/10.1049/iet-its.2009.0070
14. Abdoos M, Mozayani N, Bazzan ALC (2011) Traffic light control in non-stationary environments based on multi agent Q-learning. In: 2011 14th International IEEE conference on ıntelligent transportation systems (ITSC). IEEE, Washington, pp 1580–1585
15. Mousavi SS, Schukat M, Howley E (2017) Traffic light control using deep policy-gradient and value-function-based reinforcement learning. IET Intel Transport Syst 11:417–423. https://doi.org/10.1049/iet-its.2017.0153
16. Calvo JA, Dusparic I (2018) Heterogeneous Multi-Agent Deep Reinforcement Learning for Traffic Lights Control. In: Proceedings for the 26th Irish Conference on Artificial Intelligence and Cognitive Science (AICS), vol 2259, pp 2–13
17. Chu T, Wang J, Codeca L, Li Z (2020) Multi-agent deep reinforcement learning for large-scale traffic signal control. IEEE Trans Intell Transport Syst 21:1086–1095. https://doi.org/10.1109/TITS.2019.2901791
18. Chen C, Wei H, Xu N et al (2020) Toward a thousand lights: decentralized deep reinforcement learning for large-scale traffic signal control. Proc AAAI Conf Artif Intell 34:3414–3421. https://doi.org/10.1609/aaai.v34i04.5744
19. Yang J, Zhang J, Wang H (2021) Urban traffic control in software defined internet of things via a multi-agent deep reinforcement learning approach. IEEE Trans Intell Transport Syst 22:3742–3754. https://doi.org/10.1109/TITS.2020.3023788
20. Wang Y, Xu T, Niu X et al (2022) STMARL: a spatio-temporal multi-agent reinforcement learning approach for cooperative traffic light control. IEEE Trans Mobile Comput 21:2228–2242. https://doi.org/10.1109/TMC.2020.3033782
21. Li Z, Yu H, Zhang G et al (2021) Network-wide traffic signal control optimization using a multi-agent deep reinforcement learning. Transport Res Part C: Emerg Technol 125:103059. https://doi.org/10.1016/j.trc.2021.103059
22. Shijie W, Shangbo W (2023) A novel multi-agent deep RL approach for traffic signal control. In: 2023 IEEE ınternational conference on pervasive computing and communications workshops and other affiliated events (PerCom Workshops), pp 15–20

23. Wang T, Cao J, Hussain A (2021) Adaptive traffic signal control for large-scale scenario with cooperative group-based multi-agent reinforcement learning. Transport Res Part C: Emerg Technol 125:103046. https://doi.org/10.1016/j.trc.2021.103046
24. Mo Z, Li W, Fu Y et al (2022) CVLight: decentralized learning for adaptive traffic signal control with connected vehicles. Transport Res Part C: Emerg Technol 141:103728. https://doi.org/10.1016/j.trc.2022.103728
25. Sutton RS, Barto AG (2018) Reinforcement Learning: An Introduction, Second. The MIT Press
26. Mnih V, Badia AP, Mirza M et al (2016) Asynchronous methods for deep reinforcement learning. In: Proceedings of the 33rd ınternational conference on ınternational conference on machine learning—volume 48. JMLR.org, New York, pp 1928–1937
27. Schulman J, Levine S, Abbeel P et al (2015) Trust region policy optimization. In: Proceedings of the 32nd ınternational conference on machine learning. PMLR, pp 1889–1897
28. Schulman J, Wolski F, Dhariwal P, et al (2017) Proximal Policy Optimization Algorithms. CoRR abs/1707.06347:
29. Dhrisya K, Remya G, Mohan A (2020) Fine-grained entity type classification using GRU with self-attention. Int j inf tecnol 12:869–878. https://doi.org/10.1007/s41870-020-00499-5
30. Shaw P, Uszkoreit J, Vaswani A (2018) Self-attention with relative position representations. In: Proceedings of the 2018 Conference of the North American chapter of the association for computational linguistics: human language technologies, volume 2 (Short Papers). Association for Computational Linguistics, New Orleans, Louisiana, pp 464–468
31. Vaswani A, Shazeer N, Parmar N et al (2017) Attention is all you need. In: Advances in neural ınformation processing systems. Curran Associates, New York
32. Alegre LN (2019) SUMO-RL, https://github.com/LucasAlegre/sumo-rl
33. Wei H, Chen C, Zheng G et al (2019) PressLight: learning max pressure control to coordinate traffic signals in arterial network. In: Proceedings of the 25th ACM SIGKDD ınternational conference on knowledge discovery & data mining. ACM, Anchorage AK USA, pp 1290–1298