JST: Smart Systems and Devices
Volume 32, Issue 3, September 2022, 009-016

Exploiting Deep Reinforcement Learning for Coverage Maximization and Cost Minimization in Air Quality Monitoring Systems

Nam Duong Tran, Manh Cuong Dao, Nang Hung Nguyen, Thanh Trung Nguyen Thanh Hung Nguyen*, Phi Le Nguyen
Hanoi University of Science and Technology, Hanoi, Vietnam
Corresponding author email: hungnt@soict.hust.edu.vn

Abstract

Air-quality monitoring is highly desired in modern life, where environmental problems have become more serious. Such a task requires continuous surveillance over large urban areas, which is costly both in infrastructure and sensor resources. Hence it gives rise to Vehicular Mobile Networks (VMNs), in which mobile vehicles play the role of sensor devices and constantly monitor the area. However, with extensive constraints, the optimization of both maximizing the coverage and minimizing the sensory costs is vastly challenging. In this research, we resolve the problem in terms of a learning process. Applying deep reinforcement learning, we outperform more than 1.65% in terms of coverage, compared to common setups while remaining considerably small sensory costs in terms of sensor activation rate. We conduct extensive experiments for a better understanding of the behavior of the deep reinforcement learning model.

Keywords: Reinforcement Learning, VMNs, Energy optimization

1. Introduction

The ever-developing Internet of Things (IoT) has paved the way for many promising applications in environment surveillance tasks. Modern sensors are relatively small-sized yet highly accurate; however, still limited in coverage and computational resources. Constant activation of the sensors surely results in poor performance of the sensors as a consequence of both energy exhaustion and wasteful overlapping information. Moreover, broadly monitoring large urban areas requires an extensive amount of sensor devices which is costly. One solution is to apply sensor devices onto vehicular mobiles, creating a network of mobile sensors that traverse around the area. This helps minimize the number of sensors required. However, the problem of activation rate remains unresolved.

In this work, we attempt to optimize both the coverage in terms of the monitored area and the sensory costs in terms of sensor activation rate. Let's consider a road in an urban area, which is m (meter) in length and n (meter) in width. Many sensor-integrated mobile vehicles are constantly running along the road. We also consider the amount of time in which the air quality remains unchanged in a certain area whose radius is denoted as ϕ, we denote this duration as t_0. Based on this hypothesis, with any given point in space (x_0, y_0), the air quality at time t is the same for all points located within the circular area $(x - x_0)^2 + (y - y_0)^2 \leq \phi^2$ from the time t to $t + t_0$. The objective is to maximize the coverage and sensor activation rate in consideration of mentioned space-time constraints.

Our contributions to this work are as follows:

1) We formulate the problem into a learning problem, which we use deep reinforcement learning methods to resolve.

2) We conduct extensive experiments to both verify the efficiency of the proposed method and navigate the effect of several parameters on the performance of the proposed method.

The remaining of this article is constructed as follows. In section 2 we discuss several recent research concerning the interested field. In section 3, we give an overview of our proposal, which includes basic concepts of reinforcement learning and its applications. We then formulate our problem in the same manner in section 4 by defining essential elements for a reinforcement learning process. Next, we demonstrate the results in section 5 then the conclusion in section 6.

2. Related Works

In recent years, the development of the fifth generation of mobile communication (5G) networks [1] and mobile edge computing (MEC) technology [2] has promoted the development of intelligent vehicular mobile networks. However, when data traffic increases rapidly, the energy consumption of data processing will increase significantly, while computer resources and battery capacity are limited. Therefore, many researchers have embarked on the problem of offloading [3], scheduling [4], and resource distributing [5,6] in VMNs. In [7], Emara *et al* gave insights into applying different network architectures to the

ISSN: 2734-9373
https://doi.org/10.51316/jst.160.ssad.2022.32.3.2
Received: May 5, 2022; accepted: August 1, 2022

performance of VMNs in terms of end-to-end latency. Dong *et al* [8] applied deep reinforcement learning-based intelligent offloading scheme to resolve the problem of resource distribution in VMNs. Zhang *et al* [9] proposed a novel VMNs architecture based on SDN, in which the computational resources are distributed according to the Q-learning scheme to maximize the balance degree. Cui *et al* [10] proposed a novel offloading scheme that utilized k-nearest to select offloading interface and reinforcement learning to balance the workload throughout the network. In VMNs, it often occurs that multiple targets must be optimized simultaneously, Xu [11] proposed an evolutionary-strategy-based offloading scheme (NSGA-III) to optimize both workload distribution and latency. Xiao et al [12] also considered sensory limited energy in their work, applying deep learning to captured heat-zone and optimizing the latency based on the game theory formulation. It can be seen that the concerning problems in recent years are mainly latency and workload distribution, yet not coverage and continuous monitoring, which is the ultimate target for any sensory network.

Inspired by the above-mentioned research, we propose an optimization method based on deep reinforcement learning that focuses on the optimization of coverage and sensory costs in vehicular mobile networks.

3. Preliminaries

3.1. Reinforcement Learning

Reinforcement learning is a potential approach to real-time optimization problems. Compared to the once-famous heuristic methods, reinforcement learning offers much flexibility and lightweight computation. In combination with cutting-edge deep learning techniques, deep reinforcement learning has been utilized to achieve unprecedented excellence in many fields of interest.

In general, a reinforcement learning process consists of four essential elements that exact definition depends on the nature of the problem in which the learning process is applied:

1) *Agent*s are the source of actions. Agents make decisions according to their policy that optimizes a reward function. In many real-world scenarios, several agents interact with each other in either competition or cooperation manner. In this work, we maintain a single agent in our model.

2) *Environment* is any other factor that affects the decision made by the agent. The environment can be expressed as a set of states, and a set of mapping that maps a pair of states and actions to the next state. This mapping is often obscure and probabilistically unstable and thus difficult to know in advance. In general, an environment consists of a state-space S and a transitioning space P.

3) *Action* is a decision made by the agent. The set of actions is called the action space and is denoted as A. For each state $s \in S$, the action space at that state is denoted as $A(s)$. An action $a \in A(s)$ made by the agent affects the environment and the environment shifts to a new state s' with the probability of $p(s', a|s) \in P$. An action is taken according to the policy $p(a_t) = \pi(a_t|s_t)$.

4) *Reward* is a real scalar that shows how good the taken action is in a particular state. We denote $r_t = R(s_t, a_t)$ as the reward of the action a_t in the state s_t. The reward function R is essential for a reinforcement learning process since the reward signal guides the optimization path of the agent.

The ultimate target of the reinforcement learning process is to control the agent to make a chain of actions $\hat{A} = \{a_i\}_{i=1}^{N}$ that maximize the discounted total reward:

$$G = \sum_{t=1}^{N} \gamma^{t-1} r_t = r_1 + \gamma r_2 + \gamma^2 r_3 + \cdots + \gamma^{N-1} r_N \quad (1)$$

in which, $r_t = R(s_t, a_t)$ is the intermediate reward at time t and $\gamma \in (0,1]$ is the discount factor shows the interest degree to the future rewards. The learning process is considered finished when the agent learns a policy $\pi^* = argmax_\pi(G)$.

With such a target, most reinforcement learning methods can be categorized into two main approaches: policy-based and value-based. The policy-based approach parameterizes the policy by a vector θ and then makes an update upon this vector following the policy gradient theorem. Different methods can use different quantities as their multipliers to the gradient:

$$\begin{aligned} \nabla_\theta \pi(a_t|s_t, \theta) &= E\left[G_t \nabla_\theta \log\left(\pi_\theta(a_t|s_t)\right)\right] && \text{REINFORCE} \\ &= E[Q\nabla_\theta \log(\pi_\theta(a_t|s_t))] && \text{Q ActorCritic} \\ &= E[A\nabla_\theta \log(\pi_\theta(a_t|s_t))] && \text{Advantage} \\ &= E[\delta\nabla_\theta \log(\pi_\theta(a_t|s_t))] && \text{TD} \end{aligned}$$

On the other hand, the value-based approach tries to estimate the state-action value mapping $Q: S \times A \to R$, denoted as $Q(s_t, a_t)$ that tells the goodness of action a_t in the state s_t or the value mapping: $V: S \to R$ that tells how good a state is, concerning the overall target. After one of these mappings is well estimated, we can deduce an optimal solution for the reinforcement learning process by choosing an action that maximizes these mapping:

$$\begin{aligned} & a_t^* = argmax\left(Q(s_t, a)\right), \forall a \in A(s_t) \\ \text{or} \quad & a_t^* = argmax\left(R + \gamma V(s_t)\right), \forall a \in A(s_t) \quad (2) \end{aligned}$$

Usually, $Q(s_t, a_t)$ is favored when the joint space $S \times A$ is relatively small or computationally possible. When the joint space is infinite or relatively large, it is advantageous to utilize $V(s_t)$.

3.2. Actor-Critic Method

Each approach has its advantage in different cases. The policy-based approach is fast and flexible for both discrete and continuous space, the value-based approach is sample-efficient and stable. Actor-Critic methods are a new learning type that combines the strength of both approaches with the computational power of deep learning. The vanilla Actor-Critic architecture consists of two branches: Actor and Critic. The Actor takes a state as input s_t and returns a probability over the action space $p(\cdot\,|s_t)$ as output, meaning that the Actor plays the role of the policy π. The Critic tries to make a good estimation of the predicting reward for each action, with which the agent shifts its policy accordingly. The idea is that two models interact (or compete) with each other to achieve better performance than each separately.

There are several extensions to the vanilla Actor-Critic method, namely A2C and A3C and others. The Advantage Actor-Critic (A2C) has its Critic learn the Advantage values rather than the trivial Q values:

$$\begin{aligned} A(s_t, a_t) &= Q(s_t, a_t) - V(s_t) \\ &= E[r_{t+1} + \gamma V(s_{t+1})] - V(s_t). \end{aligned} \tag{3}$$

By learning the advantage values, the evaluation of an action is based not only on how good the action is but also on how much better it can be. The advantage of the advantage function is that it reduces the high variance of the policy networks and stabilizes the model. However, the process of learning a good Actor is still challenging due to the ineffective sampling of policy-based methods. DeepMind 2016 proposed an Asynchronous Advantage Actor-Critic (A3C) that improved this problem by introducing asynchronous multiple workers in multiple environments for better exploration of the state-action space in much less time. The agents are trained in parallel and update periodically a global network that holds shared parameters with its gradient:

$$\begin{aligned} \nabla_\theta \pi(a_t|s_t, \theta) &= E[A(s_t, a_t)\nabla_\theta \log(\pi_\theta(a_t|s_t))] \\ &\approx \left(r_{t+1} + \gamma V(s_{t+1}) - V_{s_t}\right)\nabla_\theta \log(\pi_\theta(a_t|s_t)) \end{aligned} \tag{4}$$

In this work, we leverage A2C into our proposal.

4. Proposal

4.1. Actor Critic-Based Network Modeling

Since each mobile vehicle is equipped with a sensor, itself can be considered as an agent with two decisions: activate or deactivate its sensor to monitor the surrounding air quality. However, in this setting, we embarked on the multi-agent paradigm which is rather computationally heavy as each vehicle is limited in terms of computational resources. Moreover, the number of vehicles can reach extremely large and thus makes it difficult to control the performance of the method.

For this reason, we propose using a centralized agent that takes observations of the network as input and output the activation map which is independent of the number of mobile vehicles currently on the road. By adopting this design, we can extend the proposal without the worries about the computational resources of the agent. This agent can also be viewed as a server communicating with the mobile vehicles and choosing which vehicle activates its sensor.

4.2 Actor-Critic Architecture

We employ both convolutional layers for spatial adaptation and an LSTM block for temporal analysis. We build our body network with two convolutional layers activated by ReLU functions, which help identify and analyze several spatial properties of the current state such as locations and speeds. Next, we use a fully connected layer as a preprocessor before forwarding it to the LSTM block. We intentionally use an LSTM block to handle the temporal batch of observation so that many other insights can be exploited such as the vehicle's trajectory and acceleration.

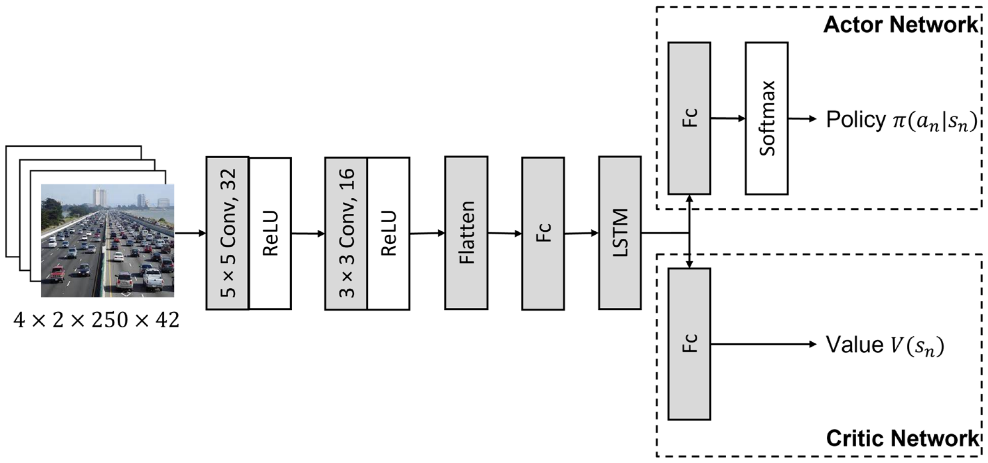


Fig. 1. Actor-Critic architecture

The output of the LSTM block is considered a spatial-temporal representation of the current state, concerning previous states. This representation is then fed into two separate branches: Actor and Critic. The output of the Actor is an $m \times n$ matrix while that of the Critic is a scalar. It is worth noting that the Actor is composed of a fully connected layer with a softmax layer while the Critic is simply a dense layer. The full demonstration of the Actor-Critic architecture is depicted in Fig. 1.

4.3. Defining Reinforcement Learning Elements

4.3.1. Environment and the state space

The environment in this problem is defined as the road and vehicles on the road. Therefore, the state space we design consists of the following components: an $m \times n$ matrix l_t that represents the location of the vehicle on the road, which can be simply understood as a binary picture of the road; and a $m \times n$ matrix c_t that demonstrate the covered map at the current time frame. The observation at time t is represented as the concatenation of the two matrices l_t and c_t. Fig. 2 demonstrates the formation of the state. The state at time t is denoted as s_t and is defined as a sequence of several observations from time $t - z$ up to t. The final state size is $(z + 1) \times 2 \times m \times n$.

The reason we design our state as mentioned is that the problem mimics a partial observation Markov decision process (POMDP). This means, that one observation at any given time is insufficient to make the optimal decision. For example, an observation of the current road does not tell the velocity of their vehicles or directions, which is critical to decide which vehicle should activate its sensor. Therefore, one practical solution is to combine the current observation with several previous observations and process them by an LSTM block.

4.3.2. Action space

Every second, a vehicle must decide whether to activate its sensor or not. From the point of view of the agent, an action is to decide which area of the road needs monitoring.

To realize this intuition, we represent an action of the agent as a probabilistic map of the road, sized $m \times n$, is denoted as $a_t = \{p_{ij}\}_{m \times n}$ in which $p_{ij} \in [0,1]$ is the probability of the location (i, j) that needs monitoring. Each vehicle on the road then self-decides to activate the sensor or not according to the deduced binary map $b_t = \{b_{ij}\}$:

$$b_{ij} = f(p_{ij}) = \begin{cases} 1, & p_{ij} \geq \varepsilon \\ 0, & p_{ij} < \varepsilon' \end{cases} \tag{5}$$

where $\varepsilon \in [0,1]$ is the threshold.

Then we have a binary matrix $b_t \in R^{m \times n}$ where 1 represents the location that needs monitoring and 0 otherwise. The vehicle at a location whose $b_{ij} = 1$ would activate its sensor.

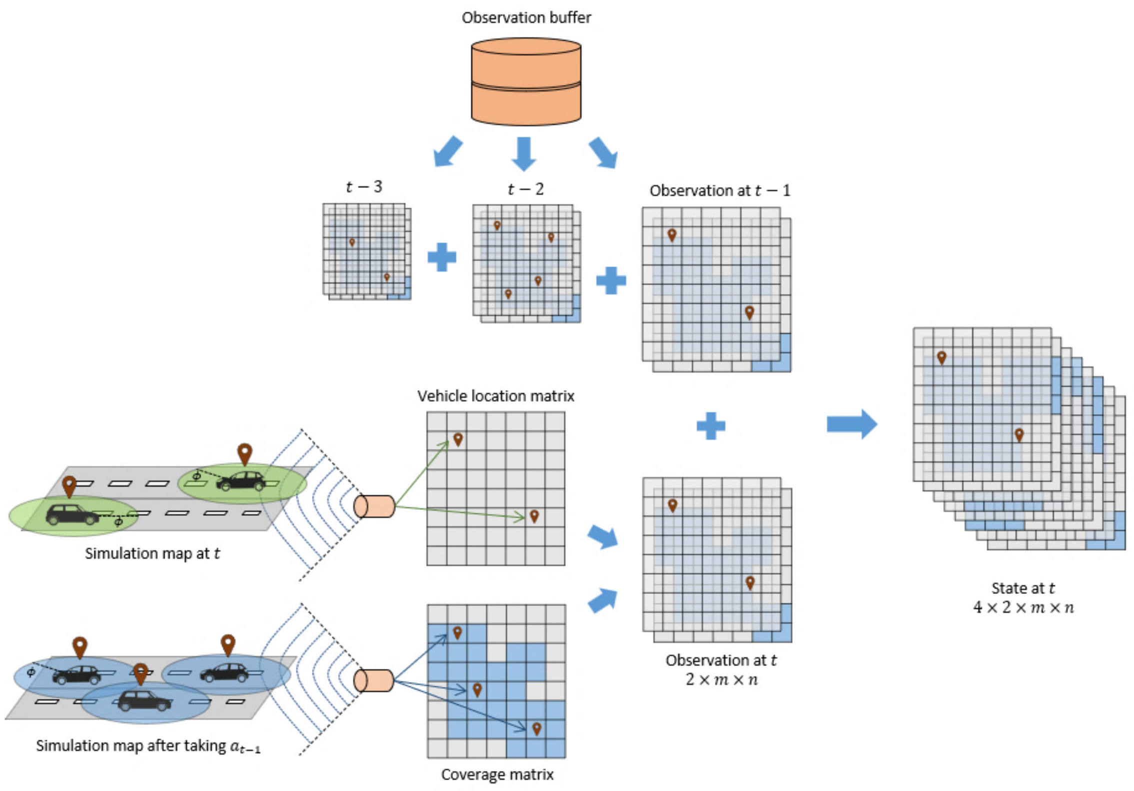


Fig. 2. The formation of the state

4.3.3. Reward

The reward signal acts as guidance for optimization in the reinforcement learning process. Thus, the design of the reward function is essential for the process to achieve high performance. A good reward function should reflect entirely the objectives that need maximizing.

In this problem, the objectives are to minimize the activation rate of the sensors while maximizing the coverage. Thus, we design our reward function based on those two elements. In detail, the reward function is defined as:

$$r(a_t) = \frac{1}{1+q^+}(\alpha R_1 - (1-\alpha)R_2) \qquad (6)$$

where:

1) $R_1 = S_{new} - S_{old} = \sum(c_{t+1} - c_t)$ is the coverage area after taking the action a_t,

2) $R_2 = (2\phi + 1)^2 o^+ - \psi$ is the overlapped area caused by performing a_t.

The notation of o^+ and q^+ represent the total vehicle on the road and the number of vehicles that activate their sensor according to the action a_t. ϕ denotes the sensory radius and ψ the previous coverage before taking a_t. The rationale behind this design is to minimize the number of vehicles participating in activating the sensors each round while maximizing the coverage area, at the same time minimizing the overlapping area. The quantities R_1 and R_2 hang in balance by a constant α, which is chosen followed by the description in Table 1.

Table 1. Configuration

Definition	Notation	Value
Road length	m	$250\,m$
Road width	n	$42\,m$
Air-quality unchanged duration	t_0	$360\,s$
Observation buffer capacity	z	3
Sensory radius	ϕ	$20\,m$
Reward trade-off factor	α	0.65

5. Evaluation

5.1. Methodology

In this section, we show the experimental results and discussion about the influence of different parameters and factors on the performance of our proposal. In detail, we first compare our proposal to several setups to verify the learning potential of the algorithm. Next, we conduct extensive experiments tuning the parameters of both network configuration and the Actor-Critic model to study their influence.

With the objective of optimizing coverage and activation rate, we are to investigate these two metrics as follows:

1) Coverage is represented as coverage degree and is defined as the ratio of the covered area over the total area (the road). This quantity is computed in a spatial-temporal manner as introduced in section 1 and described in section 4.2.1.

2) The activation rate is represented as the probability of activating the sensor at every action.

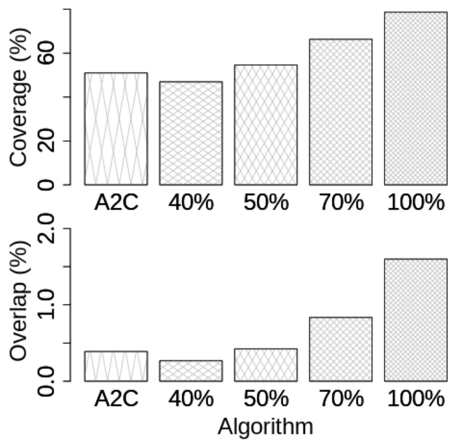
The experiments are conducted in a simulator written in Python running on a multi-processing multi-GPU machine, the deep reinforcement learning is implemented with the help of Pytorch. The seed is set constant among experiments so that the results are trustworthy.

5.2. Training Strategy

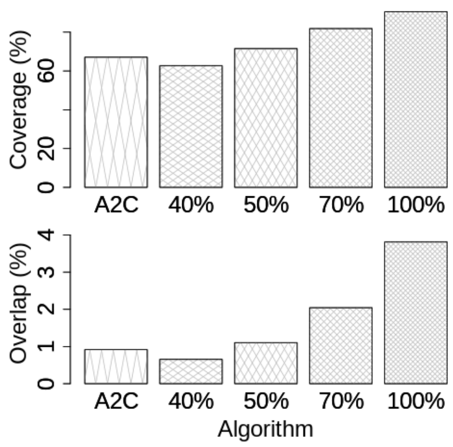
The proposal deep reinforcement learning method is trained online as follows: during the simulation, the agent continuously takes actions and receives corresponding rewards, as well as bulking the experience buffer with transitioning tuples. When the buffer is sufficiently large, the training process occurs: The transitioning tuples (experiences) are taken in consecutive batches so that our LSTM correctly learns the pattern of POMDP environmental properties such as velocity and movement direction. The actor and critic are updated accordingly. It is worth noticing that the trained experience is not discarded but restored for future rehearsals. However, we set a limit for the experience buffer, at which the buffer discards the oldest experience recordings. With this, we assure that the updating process does not cause overtime delays when updating.

5.3. Experimental Results

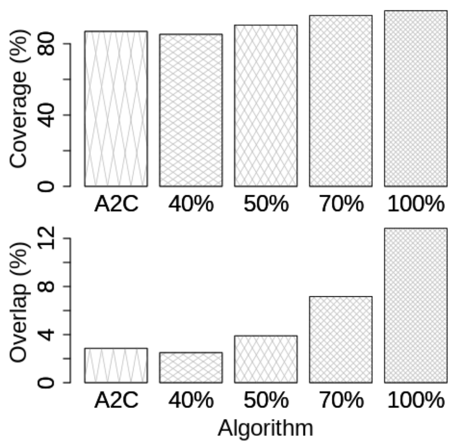
In the first experiment, we study the efficiency of the proposal. We keep the number of vehicles, the arrival interval, and other environmental factors constant while varying the sensory radius of $(3,4,6)$ meters. We compare our performance to that of the trivial setups: keep the activation rate constant at $(40,50,70,100)$ %. The results are shown in Fig. 3. In detail, when the sensory radius is set to 3, our proposal converges to the solution that activates the sensors at the rate of 46.76% and has a coverage degree of 51% and an overlap of 3.8%. This result is 45.6%, 67.1%, and 0.9% when the sensory radius is set to 4 and $42.9\%, 86.96\%$, and 2.8% when the radius is set to 6, respectively. Our proposal has decreased the activation rate throughout the experiment which is reasonable as a higher sensory radius requires less sensory activation due to the expansion of the monitoring area. Other trivial settings have competitive results in terms of coverage degree but much worse in overlap and activation rate.



(a)



(b)



(c)

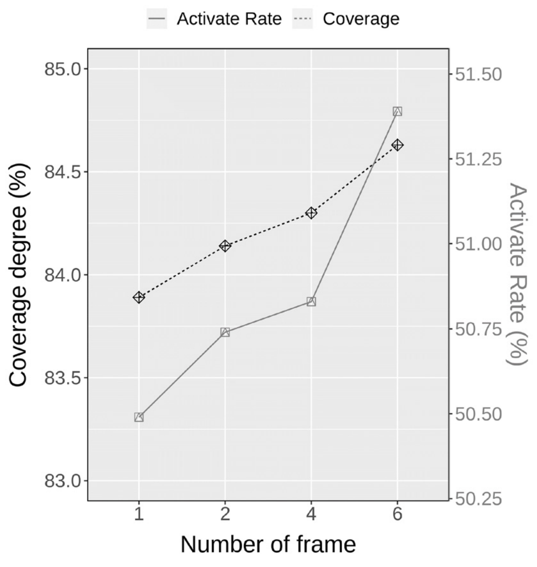
Fig. 3. Effect of Sensory Radius. a. Sensory radius is set to 3; b. Sensory radius is set to 4; c. Sensory radius is set to 6.

Next, we study the influence of inner factors on the performance of the proposal. Firstly, we increase the observation buffer capacity z from 1 to 6 and observe the change in behavior of the agent. Recall that the more consecutive observations used in the process of making a decision, the more accurate the decision becomes. However, increasing z leads to more computational requirements at the LSTM block. The result in Fig. 4a demonstrates the effect of this parameter. We can see that increasing z leads to both higher coverage degree and higher activation rate: 0.703% ($\pm$ 0.176%) on average for coverage degree and 0.52% ($\pm 0.207\%$) for activation rate. The overlap degree remains unchanged at 2.1%.

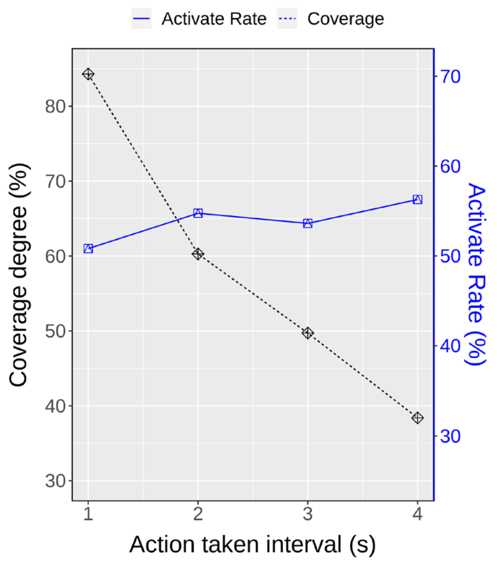
The second factor we would like to look at is the action interval of the agent, which is the amount of time between two consecutive actions. The lower this interval the more precise the decision of the agent since the change in the environment is relatively small. It can be seen in Fig. 4b that increasing the action interval leads to worse performance: higher activation rate yet lower coverage degree. In detail, when this interval is set to 1 second, the coverage degree is 84.3% and the activation rate is 50.83% whereas these numbers are 38.39% and 56.29% when the interval is 4 seconds.

The third parameter we study is the learning rate of the agent, which is also the learning rate of the Actor-Critic model. We vary this parameter from 0.05×10^{-3} to 0.5×10^{-3} and the results are shown in Fig. 4c. The results show that, with a higher learning rate, the model becomes more competent with the decrease of both coverage and activation rate, but the decreasing rate of coverage is less than that of the activation rate. In detail, the change is 0.7% in coverage and 1.7% inactivation rate. The overlap degree of these two setups is relatively equal.

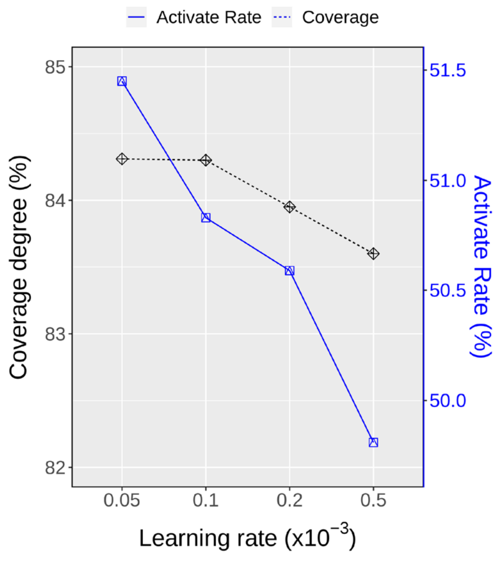
The final parameter we would like to look deep into is the hidden size of the Actor-Critic model, which is expressed through the number of nodes participating in building each hidden layer of the model. In this last experiment, we change this number from 32 to 256, which consequently increases the complexity of the model. Thus, the model often takes a longer time and more experience to attain the knowledge. It is shown in Fig. 4d that the hidden size of 64 benefits best for the performance of the Actor-Critic. In detail, when the hidden size is set at 64, the coverage degree is 84.3% and the activation rate is 50.83%. When the hidden size reaches 128 then the activation rate increases up to 52% while the coverage is roughly unchanged. At the size of 256 then the activation rate is 50.4% and the coverage degree is 83.7%. The reason for this degradation is the lack of data when performing updates upon the Actor branch.



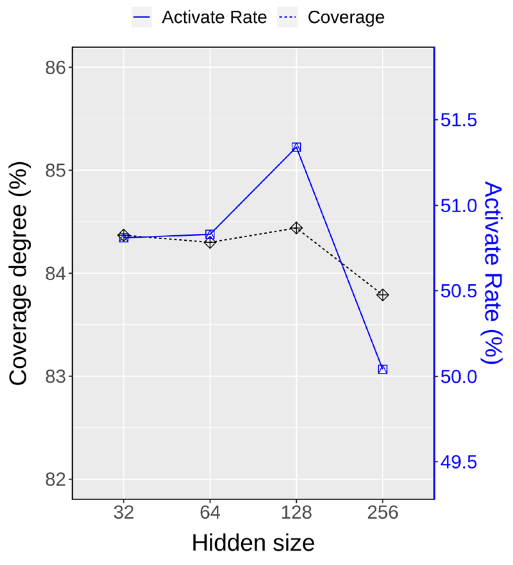
(a) The number of frames used in a state



(b) The amount of time between two consecutive action



(c) The learning rate of Actor-Critic



(d) The number of nodes in hidden layers of Actor-Critic

Fig. 4. Effect of inner parameters.

6. Conclusion and Future works

In this paper, we apply the deep reinforcement learning method to cope with the problem of coverage and sensor energy optimization in Vehicular Mobile Networks. The result shows the great potential for Deep Reinforcement Learning in terms of stability and adaptivity.

In the future, we are expanding and improving the method so that it can be applied to various scenarios where critical constraints of both sensors and the environment exist. We also consider the computational resource of the sensor as well as the vehicle for the multi-agent reinforcement learning paradigm in future works.

Acknowledgments

This work was funded by Ministry of Education and Training of Vietnam under grant number B2020-BKA-13.

References

[1] P. Zhu, J. Xu, J. Li, D. Wang, X. You, Learning-empowered privacy preservation in beyond 5G edge intelligence networks. IEEE Wireless Communications, vol. 28, no. 2, pp. 12-18, April 2021, https://doi.org/10.1109/MWC.001.2000331.

[2] J. Fu, J. Hua, J. Wen, K. Zhou, J. Li, B. Sheng, Optimization of achievable rate in the multi-user satellite IoT system with SWIPT and MEC. IEEE Trans. Ind. Inform, vol 17, no. 3, pp. 2072-2080, March 2021, https://doi.org/10.1109/TII.2020.2985157.

[3] L. Yang, H. Zhang, M. Li, J. Guo, and H. Ji, Mobile edge computing empowered energy efficient task offloading in 5G, IEEE Transactions on Vehicular Technology, vol. 67, no. 7, pp. 6398-6409, July 2018, https://doi.org/10.1109/TVT.2018.2799620.

[4] T. Q. Dinh, J. Tang, Q. D. La, and T. Q. S. Quek, Offloading in mobile edge computing: Task allocation and computational frequency scaling, IEEE Transactions on Communications, vol. 65, no. 8, Aug. 2017, pp. 3571-3584, https://doi.org/10.1109/TCOMM.2017.2699660.

[5] F. Wang, J. Xu, X. Wang, and S. Cui, Joint offloading and computing optimization in wireless powered mobile-edge computing systems, IEEE Transactions on Wireless Communications, vol. 17, no. 3, March 2018, pp. 1784-1797, https://doi.org/10.1109/TWC.2017.2785305.

[6] M. Chen and Y. Hao, Task offloading for mobile edge computing in software defined ultra-dense network, IEEE Journal on Selected Areas in Communications, vol. 36, no. 3, March 2018, pp. 587-597, https://doi.org/10.1109/JSAC.2018.2815360.

[7] M. Emara, M. C. Filippou, and D. Sabella, MEC-assisted end-to-end latency evaluations for C-V2X communications, in 2018 European Conference on Networks and Communications (EuCNC), Jun. 2018, pp. 1-9, https://doi.org/10.1109/EuCNC.2018.8442825.

[8] P. Dong, X. Wang, J. Rodrigues, Deep reinforcement learning for vehicular edge computing: An intelligent offloading system, ACM Transactions on Intelligent Systems and Technology, vol. 10, no. 6, Nov. 2019, pp. 1-24, https://doi.org/10.1145/3317572.

[9] H. Zhang, Z. Wang, K. Liu, V2X offloading and resource allocation in SDN-assisted mec-based vehicular networks, China Communications, vol. 17, no. 05, May 2020, pp. 274-291, https://doi.org/10.23919/JCC.2020.05.020.

[10] Y. Cui, Y. Liang, R. Wang, Resource allocation algorithm with multi-platform intelligent offloading in D2D-enabled vehicular networks, IEEE Access, vol. 7, 2019, pp. 21246-21253, https://doi.org/10.1109/ACCESS.2018.2882000.

[11] X. Xu, Y. Xue, X. Li, L. Qi, S. Wan, A computation offloading method for edge computing with vehicle-to-everything, IEEE Access, vol. 7, 2019, pp. 131068-131077, https://doi.org/10.1109/ACCESS.2019.2940295.

[12] Z. Xiao, X. Dai, H. Jiang, D. Wang, H. Chen, L. Yang, F. Zeng, Vehicular task offloading via heat-aware MEC cooperation using game-theoretic method, IEEE Internet of Things Journal, vol. 7, no. 3, March 2020, pp. 2038-2052, https://doi.org/10.1109/JIOT.2019.2960631.