







Adversarial retraining attack of asynchronous advantage actor-critic based pathfinding

Chen Tong¹  | Liu Jiqiang¹  | Xiang Yingxiao¹  |
Niu Wenjia¹  | Tong Endong¹  | Wang Shuoru¹  |
Li He¹  | Chang Liang²  | Li Gang³ | Chen Qi Alfred⁴

¹Beijing Key Laboratory of Security and Privacy in Intelligent Transportation, Beijing Jiaotong University, Beijing, China

²Guangxi Key Laboratory of Trusted Software, Guilin University of Electronic Technology, Guilin, China

³School of Information Technology, Deakin University, Geelong, Australia

⁴Donald Bren School of Information and Computer Sciences (ICS), University of California, Irvine, California, USA

Correspondence

Niu Wenjia and Tong Endong, Beijing Key Laboratory of Security and Privacy in Intelligent Transportation, Beijing Jiaotong University, 3 Shangyuan Village, Haidian District, 100044 Beijing, China.
Email: niuwj@bjtu.edu.cn (N. W.), edtong@bjtu.edu.cn (T. E.)

Funding information

The National Natural Science Foundation of China, Grant/Award Numbers: 61972025, 61802389, 61672092, U1811264, 61966009; National Key R&D Program of China, Grant/Award Numbers: 2020YFB1005604, 2020YFB2103800; Fundamental Research Funds for the Central Universities of China, Grant/Award Numbers: 2018JBZ103, 2019RC008; Science and Technology on Information Assurance Laboratory, Guangxi Key Laboratory of Trusted Software, Grant/Award Number: KX201902

Abstract

Pathfinding becomes an important component in many real-world scenarios, such as popular warehouse systems and autonomous aircraft towing vehicles. With the development of reinforcement learning (RL) especially in the context of asynchronous advantage actor-critic (A3C), pathfinding is undergoing a revolution in terms of efficient parallel learning. Similar to other artificial intelligence-based applications, A3C-based pathfinding is also threatened by the adversarial attack. In this paper, we are the first to study the adversarial attack to A3C, that can unexpectedly wake up longtime retraining mechanism until successful pathfinding. We also discover an attack example generation to launch the attack based on gradient band, in which only one baffle of extremely few unit lengths can successfully perform the attack. Experiments with detailed analysis are conducted to show a high attack success rate of 95% with an average baffle length of 2.95. We also

discuss defense suggestions leveraging the insights from our analysis.

KEYWORDS

A3C, evasion attack, pathfinding, reinforcement learning, retraining attack

1 | INTRODUCTION

Pathfinding, also called pathing, is responsible for finding the path from the start location to the destination in a given map. It is increasingly becoming a vital component in many real-world applications, such as famous Kiva (Amazon Robotics) warehouse systems,¹ and some autonomous aircraft towing vehicles.² In the pathfinding research family, multiagent pathfinding³ and any-angle pathfinding⁴ are attracting considerable research interests from artificial intelligence (AI), robotics, theoretical computer science, and operations research. Recently, pathfinding is moving into a new direction of unknown environment-oriented pathfinding, with the push of reinforcement learning (RL), that learns an optimal policy for agents through exhausting trials of action exploration in the unknown environment to maximize cumulative rewards. With the development of deep learning, the RL also develops new algorithms such as Q-learning-based⁵ deep Q network,⁶ deep deterministic policy gradient,⁷ and asynchronous advantage actor-critic (A3C).⁸ Due to the special parallel learning framework of A3C, A3C becomes the most promising RL in learning efficiency in a large environment. Instead of single-agent learning, A3C-based pathfinding asynchronously executes multiple agents in parallel to interact with the environment. Such parallelism makes it possible to make fast cooperative learning and push A3C-based pathfinding into broader real-world scenarios.

A3C has great potential to promote pathfinding. However, as an RL of machine learning in AI, the security of A3C whether it also opens a new door for cyber attacks is still an open issue. Recently, as described in Berkley's report,⁹ several AI security vulnerabilities have been exposed and exploited by the adversarial attack¹⁰. For instance, a special stop sign can deceive to output a recognition of the speed limit of 60 miles,¹¹ and a designed panda image can have a gibbon recognition with a great confidence.¹² Many studies have revealed adversarial attacks on RL, for example, poisoning attack¹³ and evasion attack.¹⁴ These studies highlight the RL security challenges. However, to the best of our knowledge, the former work focuses on the vulnerability analysis on nonparallel RL learning and overlook such a security issue of emerging RL mechanism. Thus, it is highly important to understand its potential security vulnerabilities of A3C parallel learning so that they can be proactively addressed before widespread deployment.

In this study, we perform the first security analysis on the emerging A3C parallel learning in real pathfinding. Our analysis focuses on the adversarial evasion attack on the retraining mechanism of RL. RL retraining actually has a nonstop connection between training process and testing process, while the retraining of supervised learning has to manually operate according to testing results. Our analysis results are expected to serve as a guideline for understanding whether and why the current implementation of A3C-based pathfinding is essentially vulnerable, as well as revealing and verifying it through high-success vulnerability exploitation. The attack requirement in our study is limited that only one baffle of extremely few unit

lengths is allowed to add into the pathfinding map by malicious attackers to cause retraining to happen and prolong the total duration of A3C-based pathfinding. The above requirement ensures that both our analysis and the discovered security problems have high practical implications.

We first analyze the implementation of A3C-based pathfinding and identify effective attacks by exhausted baffle setting in large amounts of different maps. In our analysis, we find that adversarial attacks are effective for A3C-based pathfinding: one baffle with extremely few unit lengths can nearly trigger retraining of a maximum of 247.83 min in total pathfinding. This is a surprising observation, since the A3C's policy has the ability to let agents choose alternative action to bypass the above baffle, without triggering the retraining mechanism. We find that this is due to a vulnerability of retraining trigger because of the over-trained policy learned from multiagent A3C.

Figure 1 shows the success of evasion attack by setting an attack baffle 2 in A3C usage mode. First, a multiagent training mode is used for efficient parallel training from the different start locations to the same destination (see Figure 1a). Then as what Figure 1b shows, the pathfinding starts the A3C usage mode when a converged A3C training is achieved. In such mode, the learned policy can be downloaded into any new joined agent, and any environment feedback does not be used to update the policy anymore, so as to directly utilize the optimal policy of former experiences and save unnecessary computational cost. If and only if the pathfinding duration exceeds a predefined threshold, that means the optimal policy does not work correctly, retraining will be triggered by all active agents of pathfinding in this environment until a new training converges is achieved. In pathfinding of A3C usage mode, we can see that the evasion attack of baffle 1 is not effective, because the agent chooses another action and bypasses it after the first-time failure of moving. While the evasion attack of baffle 2 has success, that the agent policy insists its moving action and eventually triggers the re-training. We find that the baffle length and its location set are key to the evasion attack for causing retraining to prolong total pathfinding duration. This is the motivation of this study,

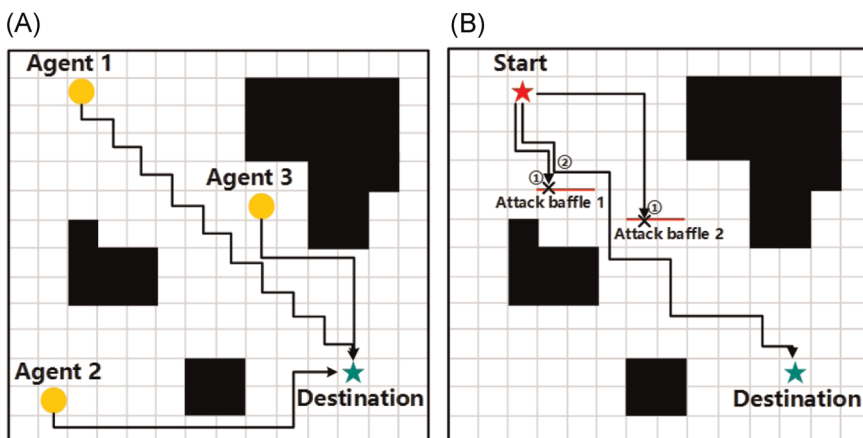


FIGURE 1 An example to show a success of evasion attack by setting an attack baffle 2 in A3C usage mode. (a) Pathfinding of A3C training mode through three agents' parallel learning; (b) pathfinding of A3C usage mode when encountering evasion attack of baffle 1 and baffle 2, respectively. A3C, asynchronous advantage actor-critic [Color figure can be viewed at wileyonlinelibrary.com]

and we prefer to first call such a successful evasion attack in A3C-based pathfinding as retraining attack. We also discuss promising defense suggestions in this study.

We also analyze the vulnerability of the retraining trigger in underlying A3C mechanism. Furthermore, considering the tension between better attack effect and less attack cost, we develop a gradient ascend- and exhaustive-based approach to vulnerability exploit, deeply revealing, and verifying the retraining attack. We summarize our contributions as follows:

1. We give the very first attempt to discover the adversarial evasion attack to the vulnerability of retraining trigger of parallel A3C learning in pathfinding. This is a novel adversarial attack to parallel RL in the specific application.
2. We propose a gradient ascend- and exhaustive-based approach to launch a retraining attack based on our vulnerability analysis.
3. We evaluate our approach empirically in real pathfinding application. Based on massive processing on four initial data sets of 100 samples, we deep further reveal and verify the retraining attack involving evaluating attack success rate, attack cost in the context of different agent numbers and retraining trigger threshold, and others.

The remainder of this paper is organized as follows. Section 2 introduces the preliminaries. Section 3 describes the retraining attack analysis. We propose corresponding exploit construction in Section 4. Experiments and detailed analysis are reported in Section 5. Section 6 gives defense suggestions. Section 7 discusses the related works. Finally, we conclude the paper in Section 8.

2 | PRELIMINARY

2.1 | A3C value functions and advantage function

The final goal of the A3C algorithm is to learn a policy π , whose input is the observation of agents (also called actors) represented as a state a , and output is a vector of action probability for any state. Then, the probability to choose a_t at t time, can use the policy

$$\pi(a_t | s_t) = p(a_t | s_t). \quad (1)$$

From task beginning to the end, agent will form a trajectory $\tau = \{s_1, a_1, s_2, a_2, \dots, s_T, a_T\}$. Then we denote τ^i as the i th trajectory.

State-action value function. Given a state s and action a , state-action value function $Q_\pi(s, a)$ is to output an expected cumulated reward (total reward) based on sampled trajectories. $Q_\pi(s_t, a_t) = \mathbb{E}(r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \dots | s_t, a_t)$. Further, we can denote it as a simple formula: $Q_\pi(s_t, a_t) = \mathbb{E}(\gamma^{t'-t} r_{t'+1}^n | s_t, a_t)$, $t' \in \{t, t+1, \dots, T\}$, in which $\gamma \in [0, 1]$ is a discount factor of reward according to Markov decision process (MDP).¹⁵ For simplicity,

$$Q_\pi(s_t, a_t) \approx \frac{1}{N} \sum_{n=1}^N \sum_{t=1}^T \sum_{t'=t}^{T_n} (\gamma^{t'-t} r_{t'+1}^n) p(a_t^n | s_t^n). \quad (2)$$

State value function. According to Bellman equation¹⁶ (dynamic programming equation), we have the state value function

$$V_{\pi}(s_t) = \sum_{a_t} \pi(a_t | s_t) Q_{\pi}(s_t, a_t). \quad (3)$$

Also, $Q_{\pi}(s_t, a_t) = \mathbb{E}[r_t + V_{\pi}(s_{t+1})]$.

Advantage function. To learn a good policy π_{θ} , it is natural to update the θ to maximize the expected total reward of all sampled trajectories, namely $\bar{R}_{\theta} = \sum_{\tau} R(\tau) p_{\theta}(\tau)$, thus a gradient ascend can be used $\theta \leftarrow \theta + \eta \nabla \bar{R}_{\theta}$. In A3C, the advantage function is defined as: $r_t^n + V_{\pi}(s_{t+1})^n - V_{\pi}(s_t^n)$, and the advantage function-based gradient is

$$\nabla \bar{R}_{\theta} \approx \frac{1}{N} \sum_{n=1}^N \sum_{t=1}^{T_n} \left(r_t^n + V_{\pi}(s_{t+1})^n - V_{\pi}(s_t^n) \right) \nabla \log p_{\theta}(a_t^n | s_t^n). \quad (4)$$

2.2 | Space of state and action in pathfinding

State space and action space are highly important for A3C working in pathfinding.

State space. S_{NM} is a state matrix for pathfinding environment of a $N \times M$ grid map, in which s_{ij} is a vector of two tuples: $\langle \text{avail}(s_{ij}), \text{coordinate}(s_{ij}) \rangle$, $\text{avail} \in \{0, 1\}$, and the state *coordinate* is denoted by (i, j) . The unit length can be defined as the distance between (i, j) and $(i \pm 1, j)$, or the one between (i, j) and $(i, j \pm 1)$.

$$S_{NM} = \begin{bmatrix} s_{00} & s_{01} & \cdots & s_{0(M-1)} \\ s_{10} & s_{11} & \cdots & s_{1(M-1)} \\ \vdots & \vdots & \ddots & \vdots \\ s_{(N-1)0} & s_{(N-1)1} & \cdots & s_{(N-1)(M-1)} \end{bmatrix}. \quad (5)$$

If there is an obstacle in s_{ij} , the *avail* is 1, otherwise *avail* = 0, thus the total state space has a size calculated as follows.

$$T_s = \sum_{n=0, m=0}^{(N-1)(M-1)} \text{avail}(s_{ij}). \quad (6)$$

Action space. For pathfinding in our work, we limit four moving actions for any state: $a(s) \in \{a_1, a_2, a_3, a_4\}$, $a_1 = \text{up}$, $a_2 = \text{down}$, $a_3 = \text{left}$, and $a_4 = \text{right}$. Thus, at t time, the state $s_t = s_{ij}$, and the action $a_t = a(s_t)$.

3 | RETRAINING ATTACK ANALYSIS

3.1 | Target multiagent A3C parallel learning

A3C utilizes a hierarchical actor-critic framework and enables asynchronous parallel training. Concretely, at the bottom layer, there are multiple agents in parallel to interact with the environment, and each agent has a pair of actor network and critic network, which is also called local actor-critic network; all agents further connect to a global actor-critic network at an upper layer. For simplicity, we still employ a common two-layer structure to discuss in the paper. As Figure 2 shows, each agent_i copies the global parameters θ before learning, then agent_i interacts with the environment_i for sampling diverse data. Each agent_i will compute the

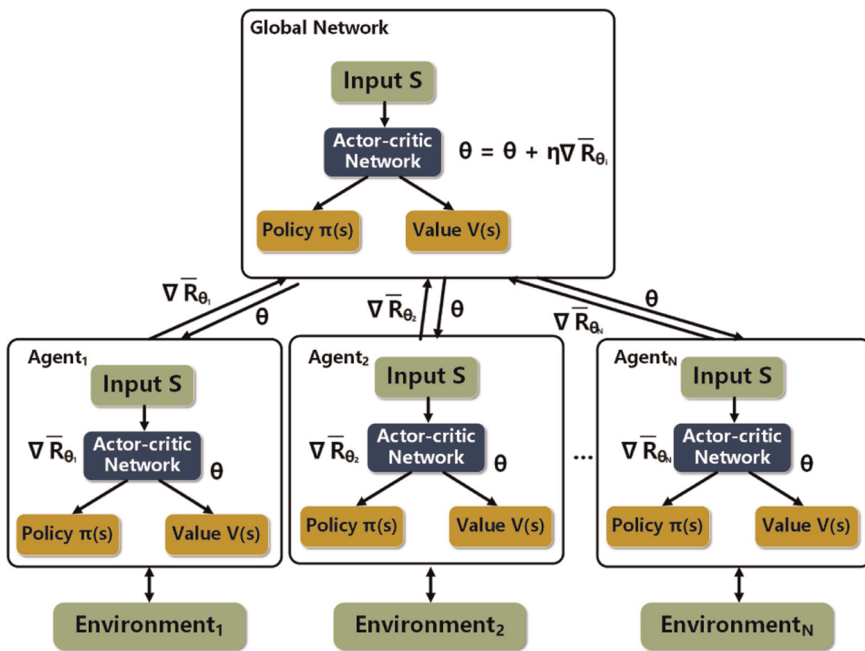


FIGURE 2 The asynchronous advantage actor-critic framework with asynchronous parallel training [Color figure can be viewed at [wileyonlinelibrary.com](https://onlinelibrary.wiley.com/doi/10.1002/mt.22380)]

gradient $\nabla \bar{R}_\theta$ after interacting, then each $agent_i$ will push the gradient to update the parameters of global network $\theta \leftarrow \theta + \eta \nabla \bar{R}_\theta$, where η is the learning rate.

In a local actor network, the actor network is responsible for obtaining an approximate state-action value function. While the local critic network is to supervise the learning of the local actor network with the state value function. The global actor-critic network does not train itself but manages cumulated updates and then cooperates with local actor-critic networks for improving training efficiency. Indeed, such training of multiple agents also has a relatively high computation cost. Thus, in a real application such as pathfinding, it is necessary to limit the training times for computational resource utilization.

3.2 | Retraining trigger for switching usage mode to training mode

For computational resource saving, in A3C-based pathfinding implementation, there are two modes to switch: training mode and usage mode.

Training mode. The training mode is used for efficient parallel training from the different start locations to the same destination. As Figure 2 shows, a complex network is utilized for iterative collaborative training by multiple agents. In simplicity, we assume that the training converges condition is a fixed large constant (e.g., 10,000) of accumulated training episode number. In RL, the term “episode” is always used to represent a time interval (length) that contains each subsequence of agent–environment interactions between initial and terminal states.

Usage mode. The A3C usage mode is started when a converged A3C training is achieved. In such mode, the learned policy can be downloaded into any new joined agent, and any

environment feedback does not be used to update the policy anymore, so as to directly utilize the optimal policy of former experiences and save unnecessary computational cost. Once the pathfinding duration exceeds a predefined threshold, that means the optimal policy does not work correctly, a retraining will be triggered by all active agents of pathfinding in this environment until a new training converges is achieved. As to the retraining trigger threshold ρ , a fixed value is to be set and different map size has different values according to expert experiences.

3.3 | Vulnerability to evasion attack

Amid a $N \times M$ finite state space with $|a(s)|$ action space, for each state $s_t \in N \times M$, there is a vector of action probability generated by policy $\pi(a_i | s_t)$, $a_i \in a(s_t)$, that is $\pi(s_t) = (\pi(a_1 | s_t), \pi(a_2 | s_t), \dots, \pi(a_{|a(s_t)|} | s_t))$.

We use $\sigma_{\pi(s_t)}$ to represent the unbiased SD of action probability vector $\pi(s_t)$ at state s_t . We have

$$\sigma(\pi(s_t)) = \sqrt{\frac{1}{|a(s_t)| - 1} \sum_{k=1}^{|a(s_t)|} (p(a_k | s_t) - \mu(\pi(s_t)))^2}, \text{ in which } \mu(\pi(s_t)) \approx \frac{1}{|a(s_t)|} \sum_{k=1}^{|a(s_t)|} p(a_k | s_t)$$

Then the t time matrix of action probability deviation can be constructed as follows:

$$M_t = \begin{bmatrix} \sigma(\pi_t(s_{00})) & \sigma(\pi_t(s_{01})) & \cdots & \sigma(\pi_t(s_{0(M-1)})) \\ \sigma(\pi_t(s_{10})) & \sigma(\pi_t(s_{11})) & \cdots & \sigma(\pi_t(s_{1(M-1)})) \\ \vdots & \vdots & \ddots & \vdots \\ \sigma(\pi_t(s_{(N-1)0})) & \sigma(\pi_t(s_{(N-1)1})) & \cdots & \sigma(\pi_t(s_{(N-1)(M-1)})) \end{bmatrix}. \quad (7)$$

In M_t , if $\sigma(\pi_t(s_{ij}))$ has a big value, then we believe that corresponding state s_{ij} is vulnerable to evasion attack, through set a new baffle to probably trigger unexpected retraining. This is because the action exploration is dependent on the action probability deviation through using the function *numpy.random.choice*($a(s), \pi(a_i | s)$). High $\sigma(\pi_t(s_{ij}))$ means an action exploration with high probability of single action choosing as an over confidence.

4 | RETRAINING ATTACK CONSTRUCTION

Our approach to construct retraining attack is outlined in Figure 3.

We can see that our approach primarily consists of three steps, that are capturing value table, producing gradient band, and baffle generation. First, we need the whole value table of A3C after converged training, then a gradient band can be fixed as a boundary constraint of baffle generation. At the last step, based on real feedback and continuous iterations, we utilize the exhaustive method to properly determine the single baffle's location and length so as to build an attack example of the retraining attack.

4.1 | Capturing the value table

The global actor-critic network does not train itself but manages accumulated updates and then cooperates with the local actor-critic networks for improving training efficiency and maintaining A3C training stability.

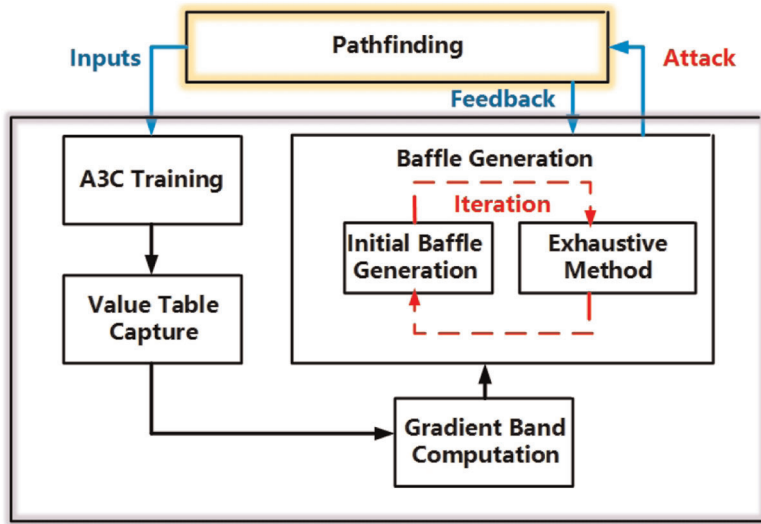


FIGURE 3 Architecture for the gradient band-based retraining attack model. A3C, asynchronous advantage actor-critic [Color figure can be viewed at [wileyonlinelibrary.com](https://onlinelibrary.wiley.com/doi/10.1002/int.2330)]

Once we get the $V_{\pi}(s_t)$ after the training converged, we can utilize $V_{\pi}(s_t)$ to estimate the value of each state s_t . For the A3C-based pathfinding in a maze with $N \times M$ grids, we can calculate its value table as matrix V , in which s_{ij} denotes corresponding state.

$$V = \begin{bmatrix} V_{\pi}(s_{00}) & V_{\pi}(s_{01}) & \cdots & V_{\pi}(s_{0(M-1)}) \\ V_{\pi}(s_{10}) & V_{\pi}(s_{11}) & \cdots & V_{\pi}(s_{1(M-1)}) \\ \vdots & \vdots & \ddots & \vdots \\ V_{\pi}(s_{(N-1)0}) & V_{\pi}(s_{(N-1)1}) & \cdots & V_{\pi}(s_{(N-1)(M-1)}) \end{bmatrix}. \quad (8)$$

V can be visualized, as both the contour map and three-dimensional surface. As shown in Figure 4, the first subgraph is the original map with a start point and a destination for pathfinding. After A3C-based pathfinding, for the V we obtained, the corresponding contour map and the three-dimensional surface can be used for revealing value distribution. In the contour map, the red line denotes the fastest way of value increment from the start position. In other words, from the start position to the destination, if we follow the gradient direction and fit each

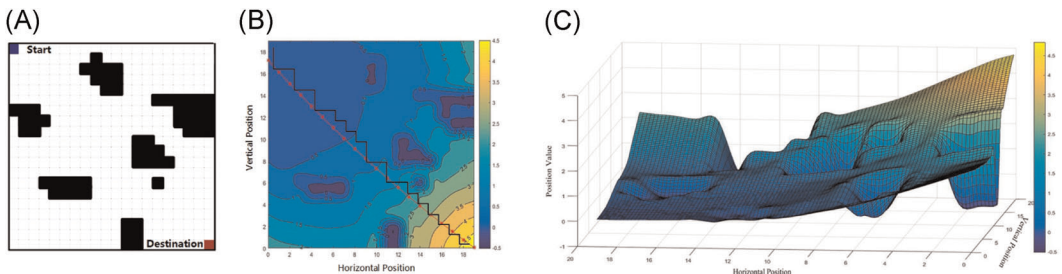


FIGURE 4 A visualization of the value table V in a well-trained asynchronous advantage actor-critic pathfinding [Color figure can be viewed at [wileyonlinelibrary.com](https://onlinelibrary.wiley.com/doi/10.1002/int.2330)]

real position into a curve, we can get such a line and we define it as a *value ascent function* f . We can see that it is close to the shortest path (see the black line). In the three-dimensional surface, the values close to the start position are relatively low, and the values close to the destination are relatively high. It is very interesting that pathfinding can be viewed as a crossing and climbing from low to high.

By summing up the common characteristics of successful attack examples, we find that the baffles added in such attack examples are roughly located within a band.

We design to approximately obtain such a band. Firstly, based on the V , we can get a value ascent function f from the start position, and then we can further generate two new functions f_{lower} and f_{upper} by moving the curve of f . Finally, the band between f_{lower} and f_{upper} is formed and we call it the gradient band. Next, we will give detailed descriptions for each step.

4.2 | Fitting the value ascent function

We capture L optimal trajectories $\tau = \{\tau_1, \tau_2, \dots, \tau_L\}$ from the start position to destination of A3C model after the training converged. Extracting states from τ and constructing a trajectory state collection $\mathbf{S} = \{S_1, S_2, \dots, S_L\}$. Thus, we can get the frequency states set from τ , which can be represented as:

$$S' = \left\{ s_{ij} \mid \sum G_k(s_{ij}) \geq \left\lfloor \frac{2L}{3} \right\rfloor (1 \leq k \leq L), \text{ or } V(s_{ij}) \geq \text{avg}(V) \right\}, \quad (9)$$

in which $G_k(s_{ij}) = 1$ if and only if $s_{ij} \in S_k$, otherwise $G_k(s_{ij}) = 0$; $V(s_{ij})$ is the value of s_{ij} obtained from value table V , $\text{avg}(V)$ means the average value of V .

Next we construct our training sample as $(x = i(s_{ij} \in S'), y = j(s_{ij} \in S'))$, and the training set size is $|S'|$. For simplicity, we use the least square (LS) method¹⁷ to fit the value ascent function. In LS, we set a linear composition of p primary functions of power functions for one input x , thus we have $A\mathbf{x} = \mathbf{y}$ as follows.

$$\begin{bmatrix} 1 & x_1 & x_1^2 & \cdots & x_1^p \\ 1 & x_2 & x_2^2 & \cdots & x_2^p \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{|S'|} & x_{|S'|}^2 & \cdots & x_{|S'|}^p \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_p \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_{|S'|} \end{bmatrix}, \quad (10)$$

in which $\mathbf{x} = [a_0, a_1, \dots, a_p]^T$, $\mathbf{y} = [y_1, y_2, \dots, y_{|S'|}]^T$. if $(A^T A)^{-1}$ exists, we can get the fitted parameters as follows:

$$\mathbf{x} = (A^T A)^{-1} A^T \mathbf{y}. \quad (11)$$

For p setting, we can determine through comparing the loss of different p by formula (12) in real training.

$$Q^2 = \sum_{i=1}^{|S'|} \left[y_i - \left(a_0 + a_1 x_i + \cdots + a_p x_i^p \right) \right]^2. \quad (12)$$

Finally, we have the value ascent function $f(x) = a_0 + a_1 x + \cdots + a_p x^p$.

4.3 | Producing the gradient band

For blocks, we extract their intersection points with all grid lines including vertical lines and horizon ones. These points are grouped into one set $E_0 = \{e_1, e_2, \dots, e_k\}$, in which $e_k = (x(e_k), y(e_k))$. Then we have the $E = E_0 \cup \{(0, 0), (X_{\max}, 0), (0, Y_{\max}), (X_{\max}, Y_{\max})\}$. Based on the f function, we can further separate E into two subset E_H and E_L , and $E = E_H \cup E_L$. $E_H = \{e_i | y(e_i) \geq f(x(e_i))\}$ and $E_L = \{e_i | y(e_i) < f(x(e_i))\}$.

Next, we calculated two minimum distances from f curve to points of E_H and E_L , respectively. Given any point e_i , we assume that there is a point r of f with coordinates $(x(r), y(r))$ that has the shortest distance to e_i . Thus, we have following equations to get r .

$$\begin{cases} f(x(r)) = y_r, \\ \mathbf{T}_r \mathbf{P}_r = 0, \end{cases} \quad (13)$$

where the tangent vector $\mathbf{T}_r = \left(-\frac{df(x(r))}{dx}, 1\right)$ at $(x(r), y(r))$ is orthogonal to the vector $\mathbf{P}_r = (x(r) - x(e_i), y(r) - y(e_i))^T$. After r is determined, we can get the shortest distance between f and e_i of E_H as follows.

$$d(e_i, f) = \sqrt{(x(r) - x(e_i))^2 + (y(r) - y(e_i))^2}. \quad (14)$$

As last, we can get the minimum distance from f to the points of E_L and E_H , respectively by formula (15).

$$D_L = \min\{d(e_i, f) | e_i \in E_L\}, \quad D_H = \min\{d(e_j, f) | e_j \in E_H\}. \quad (15)$$

As Figure 5 shows, based on the f (see the red curve), through checking the value of D_L and D_H , we can determine three cases and produce corresponding bands that we call them as gradient bands (see the green band), which cover most optimal pathfindings for converged A3C model. The detailed producing algorithm is presented as follows (Algorithm 1).

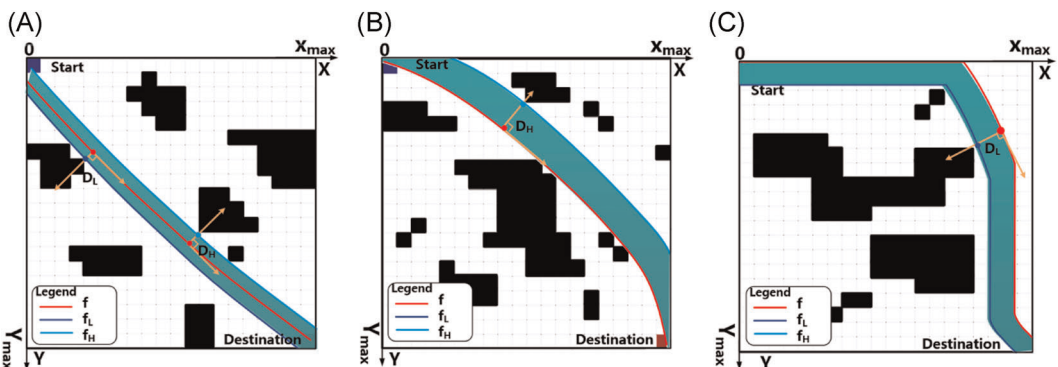


FIGURE 5 The gradient bands towards three different cases [Color figure can be viewed at [wileyonlinelibrary.com](https://onlinelibrary.wiley.com/doi/10.1002/mnt.2380)]

Algorithm 1 The gradient band production

Input: f, D_H, D_L taking the value ascent function f ,
and the minimum distance from f to the points of
 E_H and E_L , which are D_H , and D_L as input

Output: B_{gb} // an area with boundaries of f_L and f_H

1. //calculating the angle for shift direction
2. $\psi = \arctan \left| \frac{df(x(r))}{dx} \right|, \psi \in [0^\circ, 90^\circ]$
3. //Case I: $D_L \neq 0, D_H \neq 0$
4. **if** $D_L \neq 0 \ \&\& \ D_H \neq 0$ **then**
5. $f_L = f(x + D_L \times \sin\psi)$
6. $f_H = f(x - D_H \times \sin\psi)$
7. **end if**
8. //Case II: $D_L = 0, D_H \neq 0$
9. **if** $D_L = 0 \ \&\& \ D_H \neq 0$ **then**
10. $f_L = f$
11. $f_H = f(x - D_H \times \sin\psi)$
12. **end if**
13. //Case III: $D_L \neq 0, D_H = 0$
14. **if** $D_L \neq 0 \ \&\& \ D_H = 0$ **then**
15. $f_L = f(x + D_L \times \sin\psi)$
16. $f_H = f$
17. **end if**
18. **return** $B_{gb} = \{(x_i, y_i), f_L(x_i) \leq y_i \leq f_H(x_i)\}$

We can see that through ψ calculation, we can determine the shift of f , and achieve shifted f_H and f_L to form an area of band.

4.4 | Attack example generation

Given the gradient band B_{gb} in a $N \times M$ map, we can obtain an original candidate set $CA = \{a_1, a_2, \dots, a_{M-2}\}$ of attack examples to trigger retraining. For simplicity, such examples can be set as horizontal baffles with B_{gb} discretely along vertical Y axis of grid, and we have $|CA| = M - 2$, if we ignore the start and the destination of pathfinding. Thus, a baffle of attack example a_i is denoted by a horizontal segment $a_i = AB$, in which $(x(A), y(A)) = (\lfloor f_L^{-1}(y_i) \rfloor, y_i)$ and $(x(B), y(B)) = (\lfloor f_H^{-1}(y_i) \rfloor, y_i)$. We use f^{-1} to represent the inverse function of f , $\lceil \cdot \rceil$ to represent the top integral function and $\lfloor \cdot \rfloor$ to represent the low integral function. The length of baffle a_i can be calculated as follows.

$$|a_i| = |x(A) - x(B)| = \left| \left\lceil f_L^{-1}(y_i) \right\rceil - \left\lfloor f_H^{-1}(y_i) \right\rfloor \right|. \quad (16)$$

Next, we will design an automatic algorithm (See Algorithm 2) to discover the optimal retraining attack among CA set based on Definition 1 and Definition 2.

Algorithm 2 Optimal attack example discovery**Input:** $CA = \{a_1, a_2, \dots, a_{M-2}\}$ //taking the candidate set for attack examples as input**Output:** a^* //the optimal attack example

1. //initializing the optimal attack example
2. $a^* = \text{null}$
3. //traversing all candidate attack examples in CA
4. **for** $i \in \{1, 2, \dots, M - 2\}$ **then**
5. //initializing the baffle's left end
6. $x(A) = \lfloor f_L^{-1}(y_i) \rfloor$
7. **repeat**
8. //initializing the baffle's right end
9. $x(B) = x(A) + 1$
10. **repeat**
11. //determining the attack example a_i and the coordinates for a_i 's left and right end
12. $a_i = AB$
13. $(x(A), y(A)) = (x(A), y_i)$
14. $(x(B), y(B)) = (x(B), y_i)$
15. //calculating F_{effect} , F_{cost} , and F based on attack example a_i
16. $F_{\text{effect}}(a_i) = \frac{T_{\text{retraining}}^{A3C} + K \cdot T_{\text{model}}^{A3C}}{K \cdot T_{\text{model}}^{A3C}}$
17. $F_{\text{cost}}(a_i) = \frac{|a_i|}{N}$
18. $F = (F_{\text{effect}}(a_i) - \beta F_{\text{cost}}(a_i))^2$
19. **if** current F is maximum **then**
20. //recording the current optimal attack example a_i
21. $a^* = a_i$
22. **end if**
23. //determining the baffle's right end exhaustively
24. $x(B) = x(B) + 1$
25. **until** $x(B) > \lfloor f_H^{-1}(y_i) \rfloor$ //finish until baffle's right end shift out the upper boundary
26. //determining the baffle's left end exhaustively
27. $x(A) = x(A) + 1$
28. **until** $x(A) = \lfloor f_H^{-1}(y_i) \rfloor$ //finish until baffle's left end shift out the upper boundary
29. **end for**
30. **return** a^*

Definition 1 (Attack effect F_{effect}). In A3C, for baffle a_i , $F_{\text{effect}}(a_i)$ reflects its attack effect and is calculated by the time ratio of average retraining time of K agents with model-based pathfinding time to the pathfinding time, which can be calculated as:

$$F_{\text{effect}}(a_i) = \frac{T_{\text{retraining}}^{A3C} + K \cdot T_{\text{model}}^{A3C}}{K \cdot T_{\text{model}}^{A3C}}, \quad i \in \{1, 2, \dots, M - 2\}, \quad (17)$$

where $T_{\text{retraining}}^{A3C}$ reflects the retraining time to a new converge, T_{model}^{A3C} is the total time of A3C model-based pathfinding successfully from start to destination. The unit of T is

minute. Thus, $F_{effect} = 1$ means a failure attack from a_i that is unable to trigger retraining, while $F_{effect} > 1$ means a success attack and the higher F_{effect} shows better attack effect.

Definition 2 (Attack cost F_{cost}). F_{cost} is the ratio of the length of attack baffle to the grid length N , which can be calculated as:

$$F_{cost}(a_i) = \frac{|a_i|}{N}, \quad i \in \{1, 2, \dots, M-2\}, \quad (18)$$

where $0 < F_{cost}(a_i) < 1$, the smaller F_{cost} means a smaller attack cost.

At last, we can find out the optimal a_i attack with maximum attack effect and minimum attack cost by formula (19).

$$a^* = \arg \max_{a_i} (F_{effect}(a_i) - \beta F_{cost}(a_i))^2. \quad (19)$$

The β is responsible for adjusting the influence of F_{cost} . For implementation, the detailed algorithm to discover optimal a^* is shown in Algorithm 2. According to the formula (19), this algorithm tries to explore each discrete baffle of Y axis with different length, evaluating attack effect and cost to discover the optimal baffle as the real attack example to perform.

Also, we analysis the time complexity of Algorithm 2. Under the grid map of $N \times M$, an original candidate set CA can be obtained based on the gradient band produced by Algorithm 1, and $|CA| = M - 2$. To discover the optimal attack example a^* , we traverse all candidate attack examples in CA , and exhaust all possible baffle lengths within the gradient band. We assume that $w = \max_H \{[f^{-1}(y_i)] - [f_L^{-1}(y_i)] | i \in \{1, 2, \dots, M-2\}\}$, thus, the time complexity for Algorithm 2 is a level of $O(w^2 \cdot (M-2))$. We can see that the key factors for optimal attack example computation include two parts: the grid map size $N \times M$, and the gradient band's maximum width w . The smaller the grid map size is, and the more accurate the gradient band production, the time complexity for optimal attack example computation is lower.

5 | EXPERIMENTAL EVALUATION

5.1 | Experimental setup

The platform and experimental environment configuration is shown in Table 1.

Grid map data set. According to the grid map size, we build four data sets: $D_{5 \times 5}$, $D_{10 \times 10}$, $D_{15 \times 15}$, $D_{20 \times 20}$. The number of samples chosen for each D is equal to 20. For randomness, we use a common python function named *random.randint* to form a random matrix filled with 0 or 1 value, in which 0 means no block exists and 1 means a block exists. Through running 100 times of *random.randint*(0, 2, (N , N)), $N = 5, 10, 15, 20$, we have four initial data sets of 100 samples in $D_{5 \times 5}^{init}$, $D_{10 \times 10}^{init}$, $D_{15 \times 15}^{init}$, $D_{20 \times 20}^{init}$, respectively, for further fine filtering.

For the pathfinding success and block density, we perform hand filtering on initial data set: (1) filter out those samples in which there are no path from start to the destination; (2) filter out

TABLE 1 Experimental environment configuration

Platform	Experimental environment	Environment configuration
A3C	Operating system	Ubuntu 16.04.6 LTS
	CPU	Intel (R) Core (TM) i7-9700F CPU @ 3.00 GHz
	RAM	32 GB
	GPU	MSI GeForce RTX 2070 VENTUS
	Graphic memory	151MiB
	Software	Python 2.7

those samples whose block number are over 30% grid map size. Finally, we leave 20 high-quality samples to form D , for A3C algorithm training and model-based pathfinding.

Implementation. The start point of training has a coordinate (0, 0), and the end point's coordinate is $(N - 1, N - 1)$, $N = 5, 10, 15, 20$. Our pathfinding is constructed based on the program on the Github (<https://github.com/MorvanZhou/Reinforcement-learning-with-tensorflow>). In addition to developing a pathfinding training interface to invoke a parallel A3C reinforcement learning algorithm, we also encapsulate an interface to execute the well-trained A3C model for an optimal policy decision.

Both actor-learners and actor-critic controller are implemented by two different paired neural networks (called *Actor* network and *Critic* network, respectively). The *Actor* network has $2 \times 200 \times 4$ neurons from the input layer to output layer, and the *Critic* network has $2 \times 100 \times 1$ neurons.

We set the parameters as follows: the learning rate LR is set to be 0.0015, the regularization parameter $\delta = 0.001$, the value function accumulate parameter $\gamma = 0.9$ and the bias $b = 0$. We initialize the convolution kernels of *Actor* and *Critic* neural network with *random_normal_initializer*($mean = 0.0$, $SD = 0.1$), where *mean* refers to the average value and *SD* refers to the standard deviation.

We also set the trigger condition of retraining, that if an agent's moving failures are over ρ , we will stop the A3C model-based pathfinding and starts the process of multiagent A3C algorithm retraining.

Evaluation metric.

1) Attack success rate (ASR).

ASR is the ratio of the number of successful optimal attack examples to the total number of optimal attack examples, which can be calculated as follows:

$$ASR_N = \frac{\sum_{i=1}^{|D_{N \times N}|} g_i}{|D_{N \times N}|}, \quad (20)$$

where $N \times N$ is the map size, $|D_{N \times N}|$ denotes the samples' number in $D_{N \times N}$, g_i can be calculated as follows:

$$g_i = \text{sign}(F_{\text{effect}}(a^*) - 1), \quad (21)$$

in which when $F_{\text{effect}}(a^*) > 1$ the $g_i = 1$, else $g_i = 0$.

2) F_{effect} and F_{cost} .

We use $F_{effect}(a)$ to measure the attack effect caused by baffle a , and $F_{cost}(a)$ to measure the attack cost. They have been defined Section 4.4.

5.2 | Experimental results

5.2.1 | ASR and average length of optimal baffle

Table 2 presents the detailed comparison with average length of optimal baffle among $D_{N \times N}$, $N = 5, 10, 15, 20$ under different agent number (2, 3, 4, 8) and $\rho = 2, 3, 4, 5$. We can see that the maximum ASR reaches 95% when the agent number is 6 and 8, in which the minimal average baffle length is 2.95 in $D_{5 \times 5}$. The minimal ASR appears in the $D_{15 \times 15}$ when the agent number is 2 and $\rho = 5$, and its value is 0.30. Obviously, the lower threshold ρ causes an easier trigger of retraining. With the increment of ρ , the ASR decreases. For a fixed ρ , ASR increases with the increment of agent number. We find that due to more parallel training, the action

TABLE 2 ASR comparison with average length of optimal baffle among $D_{N \times N}$ under different agent number, ρ

		Grid map							
Agent number	ρ	5 × 5		10 × 10		15 × 15		20 × 20	
		ASR	Average baffle length	ASR	Average baffle length	ASR	Average baffle length	ASR	Average baffle length
2	2	0.80	2.75	0.80	4.65	0.75	6.95	0.85	8.15
	3	0.70	2.45	0.75	4.15	0.70	6.75	0.70	7.65
	4	0.60	2.25	0.55	3.25	0.55	5.85	0.60	6.85
	5	0.40	1.85	0.35	2.70	0.30	4.55	0.40	5.65
4	2	0.85	2.80	0.90	4.70	0.80	7.00	0.90	8.65
	3	0.70	2.50	0.75	4.30	0.75	6.95	0.75	7.95
	4	0.65	2.30	0.60	3.45	0.60	6.10	0.60	7.00
	5	0.45	2.05	0.40	2.85	0.35	4.75	0.40	6.00
6	2	0.95	2.95	0.95	4.85	0.90	7.25	0.90	8.60
	3	0.75	2.55	0.80	4.35	0.75	6.90	0.80	8.05
	4	0.65	2.35	0.70	3.85	0.60	6.15	0.70	7.50
	5	0.50	2.15	0.50	3.05	0.45	5.00	0.50	6.15
8	2	0.95	3.05	0.95	5.15	0.95	7.30	0.95	8.85
	3	0.80	2.70	0.85	4.65	0.8	7.10	0.90	8.20
	4	0.70	2.50	0.70	4.00	0.70	6.65	0.75	7.80
	5	0.60	2.20	0.60	3.30	0.50	5.25	0.60	6.75

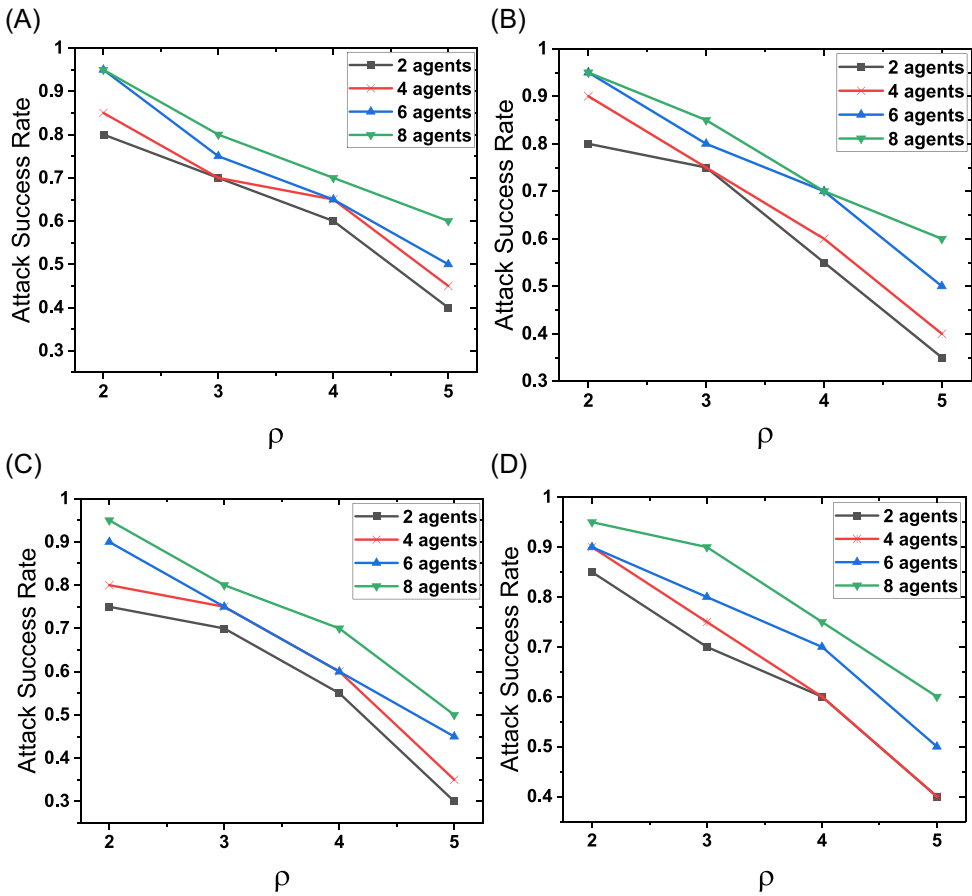


FIGURE 6 Attack success rate variation with ρ increment [Color figure can be viewed at [wileyonlinelibrary.com](https://onlinelibrary.wiley.com/doi/10.1002/mt.22380)]

probability deviation is larger in the generated gradient band, thus causing a smaller probability to let the agent choose alternative action to bypass the baffle.

As Figure 6 shows, the grid map size has little influence on ASR, while the ρ increment has observed influence on ASR. With the ρ increasing, the ASR decreases. We can see that the decrease amount of ASR when ρ from 2 to 3 is smaller than the one when ρ from 3 to 4, 4 to 5. In Figure 7, we show the distribution of average baffle length under different ASR. With the increment of grid map size, the range of baffle length is shifted from [1.85, 3.05], [2.7, 5.15], to [4.55, 7.3], and [5.65, 8.85]. The longer baffle to attack the lower trigger threshold of retraining, has a more impact on ASR and over 90% can be achieved.

5.2.2 | Average attack effect F_{effect} and attack cost F_{cost}

Table 3 shows the detailed comparison of average optimal F ($\mu(F^*)$) with average F_{effect} ($\mu(F_{effect})$) and F_{cost} ($\mu(F_{cost})$) among $D_{N \times N}$, $N = 5, 10, 15, 20$ under different agent number (2, 4, 6, 8) and $\beta = 5, 10, 15$. We can see that the maximum $\mu(F^*)$ and $\mu(F_{effect})$ reach 73539.93 and 248.83, respectively, when the agent number is 2, $\beta = 10$ in $D_{20 \times 20}$. The minimal $\mu(F^*)$ and

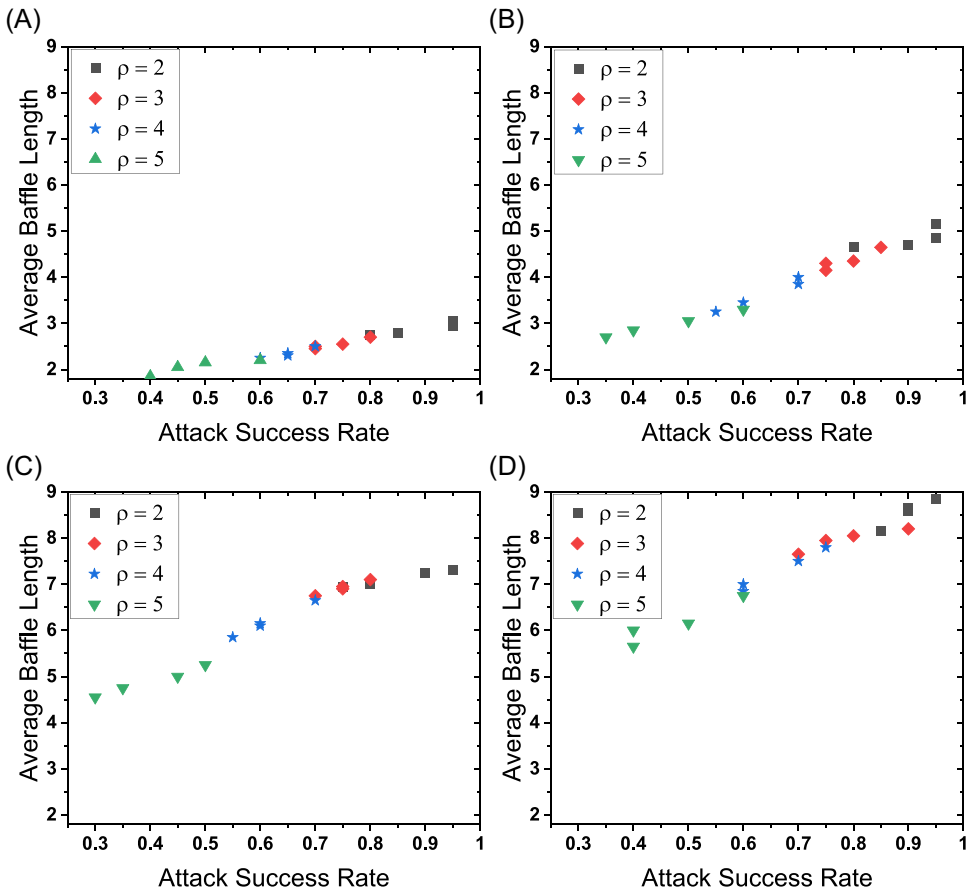


FIGURE 7 Distribution of average baffle length under different attack success rate [Color figure can be viewed at [wileyonlinelibrary.com](https://onlinelibrary.wiley.com/doi/10.1002/mt.22380)]

$\mu(F_{effect})$ appear in the $D_{5 \times 5}$ when the agent number is 8 and $\beta = 15$, and their value are 1261.86 and 35.12, respectively. The minimum $\mu(F_{cost})$ achieves 0.32 when the agent number is 2, $\beta = 15$ under $D_{15 \times 15}$. The minimal $\mu(F_{cost})$ appears in the $D_{5 \times 5}$ when agent number is 8 and $\beta = 15$.

Obviously, the larger grid map will cause an increment of retraining time. With the increment of the grid map size, the F_{effect} increases. For a fixed grid map, due to the retraining acceleration of multiple agents, thus the increment of agent number, can cause to decrease the F_{effect} .

As Figure 8 shows, the grid map size and agent number have observed influence on F_{effect} . With the increment of grid map size, the F_{effect} increases. The F_{effect} increment of 33.49%, 19.86%, and 14.52% are formed from $D_{5 \times 5}$ to $D_{10 \times 10}$, $D_{10 \times 10}$ to $D_{15 \times 15}$, $D_{15 \times 15}$ to $D_{20 \times 20}$, respectively. We can see that the decrease amount of F_{effect} when agent numbers from 2 to 4 is larger than the one when agent numbers from 4 to 6 and 6 to 8. The β shows a bigger weight on baffle size in larger grid map, especially for those A3C training based on more agents. This not only ensures a proper attack cost consideration on baffle size, but also does not affect to takes the retraining time as main component of optimal function.

TABLE 3 Average optimal F ($\mu(F^*)$) with average F_{effect} ($\mu(F_{effect})$) and F_{cost} ($\mu(F_{cost})$), among $D_{N \times N}$ under different agent number and β

Agent number		Grid map											
		5 × 5			10 × 10			15 × 15			20 × 20		
	β	$\mu(F^*)$	$\mu(F_{Effect})$	$\mu F_{(cost)}$	$\mu(F^*)$	$\mu(F_{Effect})$	$\mu(F_{cost})$	$\mu(F^*)$	$\mu(F_{Effect})$	$\mu(F_{cost})$	$\mu(F^*)$	$\mu(F_{Effect})$	$\mu(F_{cost})$
2	5	27957.20	148.86	0.47	47328.52	195.65	0.37	66849.93	237.83	0.36	73236.18	245.78	0.35
	10	31986.32	152.34	0.43	56524.64	196.24	0.34	67909.99	238.45	0.35	73539.93	247.83	0.33
	15	28369.89	151.33	0.39	48966.48	196.11	0.33	60979.63	236.12	0.32	73284.73	246.11	0.33
4	5	6471.74	74.13	0.48	12949.06	99.49	0.38	17434.37	118.42	0.39	22712.42	128.32	0.37
	10	5493.88	71.65	0.44	14340.25	99.85	0.36	17630.75	119.11	0.35	24481.94	129.44	0.36
	15	6530.86	72.45	0.40	13540.09	98.54	0.33	17366.66	118.39	0.34	21098.55	126.51	0.34
6	5	2713.03	46.37	0.50	4308.13	61.93	0.40	7268.70	75.04	0.39	18457.60	121.05	0.38
	10	2329.52	46.01	0.46	4681.56	63.71	0.37	7551.82	76.94	0.38	18277.76	122.99	0.35
	15	2232.94	47.69	0.42	4125.98	60.33	0.35	7604.47	77.12	0.35	17251.71	120.61	0.33
8	5	1295.28	35.45	0.52	2809.93	49.55	0.43	3565.07	56.72	0.41	4599.61	61.23	0.40
	10	1600.61	36.49	0.51	3071.08	51.62	0.41	3472.98	56.11	0.40	4767.40	63.45	0.38
	15	1261.86	35.12	0.49	3033.66	52.10	0.37	3757.27	58.06	0.39	4850.73	64.13	0.36

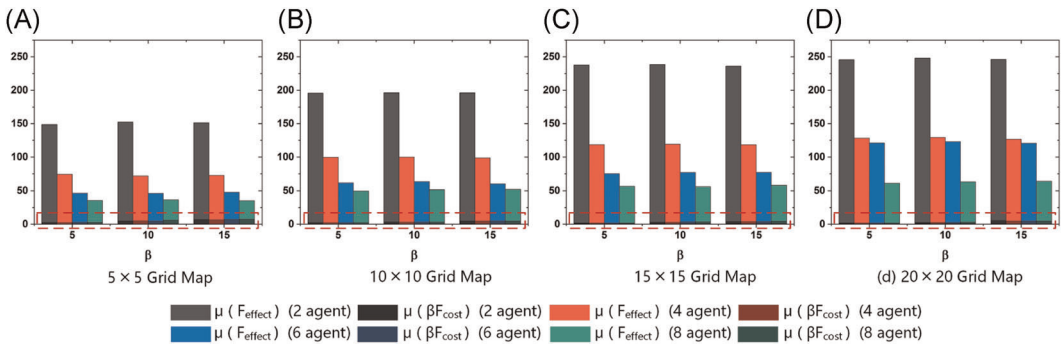


FIGURE 8 Comparison of $\mu(F_{effect})$ and $\mu(F_{cost})$ for different β [Color figure can be viewed at [wileyonlinelibrary.com](https://onlinelibrary.wiley.com/doi/10.1002/mt.2380)]

We use the box-plots (Figure 9) to report distribution of F and ASR values under different grid map. We have following observations: (1) As shown in Figure 9a, the range of F is expanded with bigger median value as well as bigger maximum value and smaller minimum value. The box size between up quartile and bottom quartile is computed, in which the biggest length is 38,510 in $D_{20 \times 20}$, and the smallest length is 17,631 in $D_{5 \times 5}$. (2) In Figure 9b, the median value of ASR varies from minimum 0.70 to maximum 0.72, having a relative small interval.

6 | DEFENSE SUGGESTIONS

As shown in our study, to proactively address the retraining attack of A3C, this section discusses defense suggestions based on the insights from our analysis.

Enabling dynamic trigger threshold of retraining. The results from the experiment show that small threshold ρ (such as $\rho = 2, 3$) can cause a retraining trigger through adversarial baffle setting, while usually in normal condition without above evasion attack, such small threshold works well for ensuring A3C model's performance through timely A3C model's by retraining parameter updating. However, it is not plausible to set a ρ initially with a big value, which will make it difficult to start the retraining mechanism, causing the lack of robustness towards exception of the running environment. Moreover, such a threshold should be protected as an important hyperparameter for

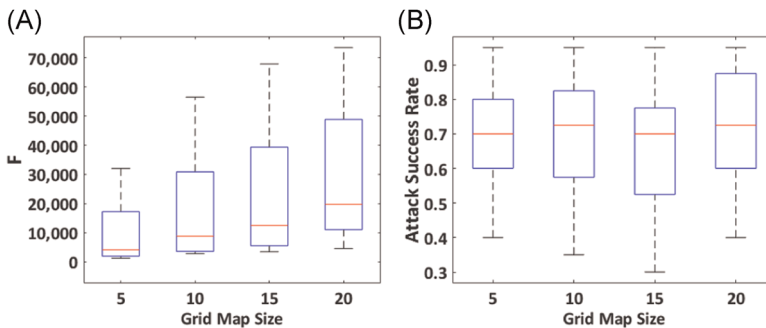


FIGURE 9 Value distribution of F and ASR in different $D_{N \times N}$ [Color figure can be viewed at [wileyonlinelibrary.com](https://onlinelibrary.wiley.com/doi/10.1002/mt.2380)]

security. If frequent retraining triggers are detected, a corresponding alert should be generated for further verification that whether to dynamically adjust the current threshold.

Proper agent amount for A3C training. Through multiagent parallel training, A3C has a high learning efficiency. Although fewer agents mean lower cost of computing resource consuming, it also brings a risk of bigger attack effect F_{effect} from retraining attack. However, more agents based learning will increase the threat of retraining attack to the A3C model, because of the larger action probability deviation of states within the gradient band. This is the art of trade-off between attack threat towards retraining attack and attack effect F_{effect} after an attack. To balance, we suggest taking a middle and proper agent amount for A3C training.

Enriching agent's ability of perception and feedback. As in most RL applications, the agent has limited perception ability that it is only aware of the macrolevel failure, such as the state's unreachableness. While the detailed reason and fine feedback of failure type depend on different applications. For A3C-based pathfinding, we suggest supporting computer vision-based obstacles sensing to distinguish common environment obstacles and low-cost attack baffles. In addition, the feedback of action failure from blocking and the one from the agent itself should be distinguished for timely detection on retraining attack.

7 | RELATED WORK

7.1 | Adversarial examples in evasion attack

A lot of studies have investigated the adversarial attack in different scenarios. These studies highlight the security challenges and the severe consequences, in which most efforts can be thought of making almost invisible or inaudible perturbations into original inputs for data spoofing. These inputs are primarily distributed to the types of image and audio. For example, Kurakin et al.¹¹ showed the adversarial image examples can perform deception with 87% success rate from the printer's picture to the real-world photo. A face image was modified such that it can change the gender prediction, whereas that face's biometric utility remained intact.^{18,19} Eykholt et al.²⁰ introduced adversarial attacks from physical road signs, which seems no abnormality as we see. For audio, Carlini et al.²¹ produced adversarial voice commands that people can not hear. Carlini et al.²² constructed targeted adversarial audio examples under any given audio waveform by adding audio noises. Lately, black-box targeted attacks were performed by simulating transformations caused by playback or recording in the physical world.^{23,24} Taori et al.²⁵ showed a 89.25% targeted attack similarity in audio with 35% targeted attack success rate.

Above adversarial example belongs to an implicit adversarial example with the invisible or inaudible data spoofing. In comparison, a more recent study tried to construct dominant adversarial examples with more obvious perturbations. Elsayed et al.²⁶ directly embedded a obvious whole image as noise into the original image. Apart from image and audio, some recorded network features were also modified to perform adversarial attacks against intrusion detection.²⁷ Our paper is the first study that exposes concrete data spoofing of dominant adversarial example by adding short baffle in pathfinding application. Compared to implicit image or audio attack example, baffle-based dominant attack example can be easily set, and thus is able to bring practical and severe consequence as well, which should not be overlooked.

7.2 | RL security

Before our study, the vulnerability exploits of AI models especially in machine learning subdomain, research to let machine trained to recognize patterns, have been studied a lot. These efforts highlight and reveal valuable vulnerabilities to those famous machine learning models, such as support vector machine,²⁸ deep natural networks (DNNs),²⁹ convolutional neural network,¹² recurrent neural network,³⁰ and long short term memory.³⁰

There are many studies that explore the security problem in reinforcement learning. Behzadan and Munir et al.³¹ discovered that the self-driving platooning vehicles can collide with each other when their observation data are manipulated. Drones equipped with RL techniques can be commanded to collide to a crowd or a building.^{32,33} Everitt et al.³³ and Wang et al.³⁴ investigated RL algorithms under corrupted reward signals. Lin et al.³⁵ and Behzadan and Munir³⁶ focused on deep RL which involves DNNs for function approximation. Huang and Zhu et al.³² studied RL under malicious falsification on cost signals and introduced a quantitative framework of attack models to understand the vulnerabilities of RL. Ma et al.³⁷ focused on security threats on batch RL and control where the attacker aims to poison the learned policy.

In comparison, we are the first to study RL security on parallel training-based A3C under the experimental scenario of pathfinding. Moreover, this does not belong to a kind of poisoning attack but an evasion attack, which is much more realistic in a black-box attack. Compared to the policy attack in related work, we also reveal an attack on retraining, an important mechanism for integrating multiagent A3C training and single-agent A3C model usage. To our best knowledge, this is the very first work.

To summary, as Table 4 shows, the experimental scenarios among state-of-the-art works in the field of adversarial attack mainly including computer vision, audio, intrusion detection, control system, Atari game, tabular certainty-equivalence, and linear quadratic regulator. And our work gives the very first attempt to reveal the vulnerability of A3C under the experimental scenario of pathfinding. Moreover, in former work on aiming reinforcement learning mostly focuses on poison attack. While the evasion attack developed in this study is much more realistic in the black-box attack. In addition, our work proposes a kind of dominant adversarial example, which can be set easier, and is able to bring a more practical consequence. In comparison, we give the very first attempt to reveal the vulnerability of the retraining trigger of A3C in the pathfinding. Moreover, we propose a kind of evasion attack, namely retraining attack, which poses a threat to the effectiveness of A3C under the scenario of pathfinding. However, the retraining attack proposed in this paper focuses on the vulnerability of parallel learning A3C in pathfinding only, thus there exist certain limitations on the available application scenarios of this approach. In the future, we plan to develop a more sophisticated attack approach, which mainly focuses on the design vulnerability of the parallel learning algorithm itself. So that our experiment can be implemented on a wider range of application scenarios.

8 | CONCLUSION

In this study, we perform the first attempt to discover a retraining attack to the vulnerability of retraining trigger of parallel A3C learning in pathfinding. Targeting such vulnerability exploitation, we propose a gradient ascent- and exhaustive-based approach to launch a retraining attack. We implement an A3C-based pathfinding application as our experimental environment, and the massive experiments with a different combination of agent number, retraining trigger

TABLE 4 The comparison of target model, experimental scenario, and type of adversarial example and attack among state-of-the-art works in the field of adversarial attack

Work	Target model	Experimental scenario	Example's type	Attack's type
Mirjalili et al. ¹⁹	DNN—IntraFace	Computer vision—Gender prediction for face image	Implicit	Evasion attack
Eykholt et al. ²⁰	CNN—LISA>SRB	Computer vision—Physical road sign recognition	Implicit	Evasion attack
Taori et al. ²⁵	DRN—Speech recognition system	Audio—Audio recognition	Implicit	Evasion attack
Elsayed et al. ²⁶	DNN—Google Inception Net v3	Computer vision—Image classification	Dominant	Evasion attack
Corona et al. ²⁷	Intrusion detection system	Intrusion detection	Dominant	Evasion attack
Huang et al. ³²	Q-Learning	Control system—Water reservoir control	Implicit	Poison attack
Lin et al. ³⁵	DQN	Atari game	Implicit	Poison attack
Ma et al. ³⁷	Batch reinforcement learning	Tabular certainty-equivalence and Linear quadratic regulator	Implicit	Poison attack
Our work	A3C	Pathfinding	Dominant	Evasion attack

Abbreviations: A3C, asynchronous advantage actor-critic; CNN, convolutional neural network; DDN, deep natural network; DRN, deep recurrent network; DQN, deep Q-network.

threshold, and other parameters show the effectiveness of our approach. Our approach shows a high attack success rate of 95%, and one baffle with extremely few unit lengths can nearly trigger retraining of maximum of 247.83 min in total pathfinding. We also give promising suggestions to prevent retraining attack.

This study severs as a first step to take security analysis on emerging A3C parallel learning in real pathfinding. It is expected to inspire a series of follow-up studies, including but not limited to (1) more extensive evaluation in different A3C based applications and (2) more concrete defense design and implementation.

ACKNOWLEDGEMENTS

The National Natural Science Foundation of China (61972025, 61802389, 61672092, U1811264, and 61966009), the National Key R&D Program of China (2020YFB1005604, 2020YFB2103800), the Fundamental Research Funds for the Central Universities of China (2018JBZ103 and 2019RC008), Science and Technology on Information Assurance Laboratory, Guangxi Key Laboratory of Trusted Software (KX201902).

ORCID

Chen Tong  <http://orcid.org/0000-0003-3851-2135>

Liu Jiqiang  <http://orcid.org/0000-0003-1147-4327>

Xiang Yingxiao  <http://orcid.org/0000-0002-8679-7000>

Niu Wenjia  <http://orcid.org/0000-0003-4706-4266>

Tong Endong  <http://orcid.org/0000-0003-0348-2108>

Wang Shuoru  <http://orcid.org/0000-0002-8003-6745>

Li He  <http://orcid.org/0000-0002-4073-5816>

Chang Liang  <http://orcid.org/0000-0002-7262-4707>

REFERENCES

1. Wurman PR, D'Andrea R, Mountz M. Coordinating hundreds of cooperative, autonomous vehicles in warehouses. *AI Magazine*. 2008;29(1):1-9.
2. Morris R, Pasareanu CS, Luckow KS, et al. Planning, scheduling and monitoring for airport surface operations. In: *AAAI Workshop: Planning for Hybrid Systems*. AAAI; 2016:608-614.
3. Andreychuk A, Yakovlev K, Atzmon D, Stern R. Multi-agent pathfinding (mapf) with continuous time. In: *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*. 2019:39-45.
4. Harabor DD, Grastien A. An optimal any-angle pathfinding algorithm. In: *ICAPS*. 2013:308-311.
5. Watkins C. *Learning from Delayed Rewards*. Cambridge University; 1989.
6. Mnih V, Kavukcuoglu K, Silver D, et al. Playing atari with deep reinforcement learning. In: *Neural Information Processing Systems*. 2013:1-9.
7. Lillicrap TP, Hunt JJ, Pritzel A, et al. Continuous control with deep reinforcement learning. In: *International Conference on Learning Representations*. 2016:168-174.
8. Mnih V, Badia AP, Mirza M, et al. Asynchronous methods for deep reinforcement learning. In: *International Conference on Machine Learning*. 2016:1928-1937.
9. Ren T. Berkeley released ai system challenge report, 2017.
10. Szegedy C, Zaremba W, Sutskever I, et al. Intriguing properties of neural networks. In: *International Conference on Learning Representations*. 2013:1-10.
11. Kurakin A, Goodfellow I, Bengio S. Adversarial examples in the physical world. In: *International Conference on Learning Representations*. 2017:1-14.
12. Goodfellow IJ, Shlens J, Szegedy C. Explaining and harnessing adversarial examples. In: *International Conference on Learning Representations*. 2015:1-11.

13. Bojchevski A, Günnemann S. Adversarial attacks on node embeddings. In: *International Conference on Machine Learning*. 2018:695-704.
14. Kurakin A, Goodfellow I, Bengio S. Adversarial examples in the physical world. In: *International Conference on Learning Representations*. 2016:14254-14263.
15. Bellman R. A markov decision process. *Journal of Mathematical Mechanics*. 1957:679-684.
16. Bellman R. Dynamic programming. *Science*. 1966;3731(153):34-37.
17. Björck Å. *Numerical Methods for Least Squares Problems*. Vol 66. SIAM; 1996.
18. Liu Z, Luo P, Wang X, Tang X. Deep learning face attributes in the wild. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2015:3730-3738.
19. Mirjalili V, Ross A. Soft biometric privacy: Retaining biometric utility of face images while perturbing gender. In: *2017 IEEE International joint conference on biometrics (IJCB)*. IEEE; 2017:564-573.
20. Eykholt K, Evtimov I, Fernandes E, et al. Robust physical-world attacks on deep learning visual classification. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. IEEE; 2018:1625-1634.
21. Carlini N, Mishra P, Vaidya T, et al. Hidden voice commands. In: *USENIX Security Symposium*. 2016:513-530.
22. Carlini N, Wagner D. Audio adversarial examples: Targeted attacks on speech-to-text. In: *2018 IEEE Security and Privacy Workshops (SPW)*. IEEE; 2018:1-7.
23. Alzantot M, Balaji B, Srivastava M. Did you hear that? adversarial examples against automatic speech recognition. In: *31 st Conference on Neural Information Processing System*. 2017:1-6.
24. Yakura H, Sakuma J. Robust audio adversarial example for a physical attack. In: *International Joint Conference on Artificial Intelligence*. 2019:5334-5341.
25. Taori R, Kamsetty A, Chu B, Vemuri N. Targeted adversarial examples for black box audio systems. In: *2019 IEEE Security and Privacy Workshops (SPW)*. IEEE; 2019:15-20.
26. Elsayed GF, Goodfellow I, Sohl-Dickstein J. Adversarial reprogramming of neural networks. In: *International Conference on Learning Representations*. 2019:1-14.
27. Corona I, Giacinto G, Roli F. Adversarial attacks against intrusion detection systems: Taxonomy, solutions and open issues. *Information Sciences*. 2013;239:201-225.
28. Biggio B, Nelson B, Laskov P. Support vector machines under adversarial label noise. In: *Asian Conference on Machine Learning*. 2011:97-112.
29. Nguyen A, Yosinski J, Clune J. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. IEEE; 2015:427-436.
30. Papernot N, McDaniel P, Swami A, Harang R. Crafting adversarial input sequences for recurrent neural networks. In: *MILCOM 2016–2016 IEEE Military Communications Conference*. IEEE; 2016:49-54.
31. Behzadan V, Munir A. Adversarial reinforcement learning framework for benchmarking collision avoidance mechanisms in autonomous vehicles. *IEEE Intelligent Transportation Systems Magazine*. 2019.
32. Huang Y, Zhu Q. Deceptive reinforcement learning under adversarial manipulations on cost signals. In: *International Conference on Decision and Game Theory for Security*. Springer; 2019:217-237.
33. Xu Z, Zhu Q. A cyber-physical game framework for secure and resilient multi-agent autonomous systems. In: *2015 54th IEEE Conference on Decision and Control (CDC)*. IEEE; 2015:5156-5161.
34. Wang J, Liu Y, Li B. Reinforcement learning with perturbed rewards. In: *AAAI*. 2020:6202-6209.
35. Lin Y-C, Hong Z-W, Liao Y-H, Shih M-L, Liu M-Y, Sun M. Tactics of adversarial attack on deep reinforcement learning agents. In: *Processdings of the Twenty-Sixth International Joint Conference on Artificial Intelligence (IJCAI-17)*. 2017:3756-3762.
36. Behzadan V, Munir A. The faults in our pi stars: Security issues and open challenges in deep reinforcement learning. arXiv preprint arXiv:1810.10369. 2018:1-16.
37. Ma Y, Zhang X, Sun W, Zhu J. Policy poisoning in batch reinforcement learning and control. In: *Advances in Neural Information Processing Systems*. 2019:14570-14580.

How to cite this article: Tong C, Jiqiang L, Yingxiao X, et al. Adversarial retraining attack of asynchronous advantage actor-critic based pathfinding. *Int J Intell Syst*. 2021;36: 2323-2346. <https://doi.org/10.1002/int.22380>