Final Report for Mobile Internet

Hongyi Guo
516030910306
guohongyi@sjtu.edu.cn

Jikun Chen
516030910303
cjk7989@sjtu.edu.cn

Abstract

We consider the connectivity determination of a network under the risk of adversarial models. The task is to design a defense strategy to predict and protect the edges that is most likely to be attacked by an attacker. In this paper, we propose a framework based on reinforcement learning, in which we simultaneously train an attacker model and a defender model. The models generate actions against each other, which is similar to the training process of generative adversarial networks (GAN). To test the efficiency of our method, we also implement a rule-based method and a stochastic method. Experiments show that the RL-based model performs better than the other two methods. Our codes are released on github[1].

1 Introduction[2]

In this paper we study connectivity determination under the risk of adversarial model. Consider a network as a connected digraph, we assume that an attacker can delete a number of edges according to certain strategies, so that some vertices will lose their connections. This type of attack may happen in a real scene. What we should do is to predict the edges that is most likely to be deleted by an attacker, and protect them in advance.

In a simple problem, an adversarial attacker attempts to destroy the connectivity of a given source-destination pair. For the attacker, the most efficient way is deleting the min-cut of the graph. However, it is also easy to defend due to the above property. So we would like to study a harder problem, which will be introduced as follows.

1.1 Problem Description

Firstly, we will redefine connectivity. Consider the scene that in a network $G = (V, E)$ with a given source point $src \in V$. Define connectivity as the number of points reachable by the source point divided by the total number of points other than the source, which can be written as

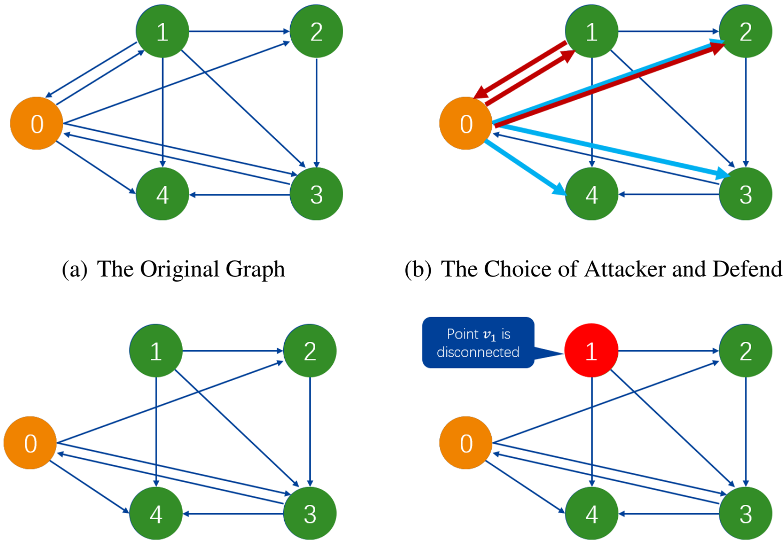
$$Connectivity = \sum_{v \in V, v \neq src} \frac{1_{connected(src,v)}}{|V| - 1}$$

In this scene, an attacker is trying to reduce the connectivity by deleting K edges, while a defender can choose K edges and protect them from attacks. The attack on the edge that has been protected will fail. Figure 1 gives an example of a round of attack and defense.

When the attacker and the defender make their decisions, they don't know each other's choice. Since an adversary can attack the network in a variety of ways, it is important for the defender to be robust enough. Our goal is to develop stronger defender that can protect the connectivity.

[1] Source code address: https://github.com/gohsyi/secure_connectivity

[2] This section is written by J. Chen



(a) The Original Graph

(b) The Choice of Attacker and Defender

(c) The Graph After Deletion

(d) The Reduction of Connectivity

Figure 1: A round of attack and defense. First, the network is represented by a directed graph G with $N(=5)$ vertices and $M(=11)$ edges. The source point is v_0 by default, and $K = 3$. Note that in the original graph, the source point is connected to all other four vertices, the connectivity of the network is 1. In a round of attack and defense, edges selected by the defender are colored blue, edges selected by the attacker are colored red. After that, two edges without protection are deleted. Finally, the source point loses connection with v_1. The connectivity of the network becomes 0.75.

1.2 Our Work

We propose a framework based on reinforcement learning to build a strong defender. Inspired by GAN, we model an attacker and a defender as the same time. In the training process, a random graph is created from the environment as the observation. Then it is inputted into the attacker model and the defender model. The two models are trained simultaneously, using a2c algorithm. The attacker's action is to select some edges to delete. The defender's action is to select some edges that can't be deleted. At the end of a training step, the environment detect the connectivity of the graph. If l points are disconnected from the source point after the attack, the attacker will get $\frac{l}{N-1}$ reward, and the defender will get $-\frac{l}{N-1}$ reward as a punishment.

To test the efficiency of our method, we implement another two methods for comparison. One is called rule-based method, where the attacker and the defender generate actions according to a certain strategy. The other is called stochastic method, where the two models select the edges randomly. We set a series of experiments to compare the ability of the above methods, and find that the RL-based method performs better than the others in both attack and defense.

2 Related Work[3]

Our work is related to a number of previous work. Our problem scenarios is designed inspired by work on connectivity of dynamic graphs, such as shortest path problem under adversarial attacks, matchings in bipartite random graphs, and dynamically evolving random graphs. Our solution is based on reinforcement learning and generative adversarial networks.

[3] This section is written by J. Chen

2.1 Work on Connectivity of Dynamic Graphs

Connectivity of dynamic graphs has been studied for more than twenty years. Some researchers studies shortest path problem. [1] considers the shortest path routing (SPR) of a network with stochastically time varying link metrics under potential adversarial attacks. Given an directed acyclic graph where link weights follow some unknown stochastic distributions and can be attacked by adversaries, the task is to find the shortest path between a source-destination pair. The authors formulate it as a multi-armed bandit (MAB) problem, and use EXP3 algorithm to solve it. [2] studies loop-free stochastic shortest path problems. The authors consider online learning in it, and propose an algorithm called "follow the perturbed optimistic policy". The Algorithm has a regret bound against an arbitrary sequence of reward functions but for the case when the Markovian dynamics is unknown. [3] is about how to find the maximum matching in bipartite graphs after an adversary deleted some edges, and [4] studies adversarial deletion in a dynamically evolving random graph. Beside the work to solve the problems, both of them discuss interesting properties of the graphs under adversarial attacks.

Although the above studies focus on different problems, all of them are related to the connectivity of dynamic graphs. This is benefit to our work. For example, [1] design a scene how does an adversarial attacker destroy the connectivity of a graph. What's more, both of the studies on shortest path problem is solved with reinforcement learning. This brings great inspiration to us.

2.2 Reinforcement Learning

In a standard reinforcement learning [5]ite process, a model interacts with an environment. At time step t, the agent receives a state s_t and generate an action a_t. The environment receive the pair $< s_t, a_t >$, then give the model the next state s_{t+1} and a reward r_t. r_t is used to train the model. This process continues until the model reaches a terminal state. The goal is to maximize the return $R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k}$.

A2C [6] is a conceptually simple and lightweight framework for deep reinforcement learning that uses asynchronous gradient descent for optimization of deep neural network controllers. In this paper, we implement the RL-based method via A2C.

2.3 Generative Adversarial Networks

The generative adversarial networks (GAN) [7] framework is used to estimate and optimize generative models via an adversarial process. It has two models: a generative model G that captures the data distribution and generate a sample, and a discriminative model D that estimates the probability that a sample came from the training data rather than G. The models are trained simultaneously, as shown in Figure Figure 2. Finally, G can generate samples similar to real ones, and D becomes a good discriminator.

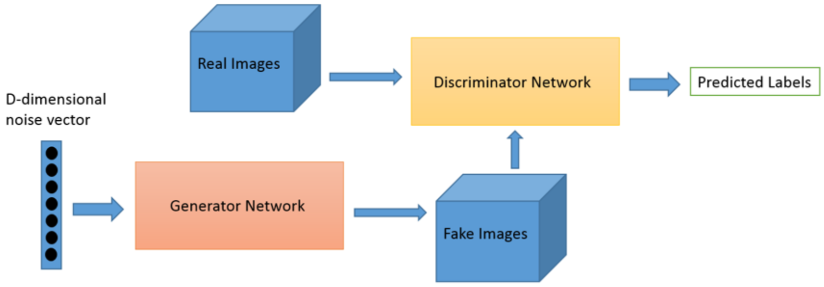
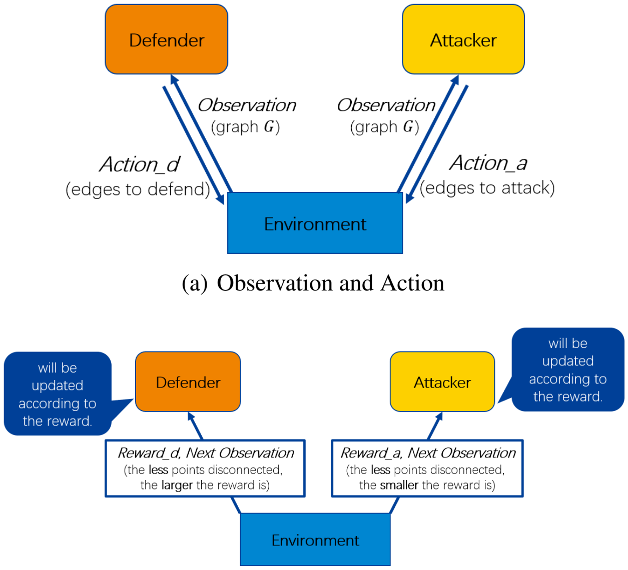


Figure 2: Structure of GAN

3 Methodology[4]

3.1 Framework


Defender
Attacker
Observation (graph G)
Observation (graph G)
Action_d (edges to defend)
Action_a (edges to attack)
Environment
will be updated according to the reward.
Defender
Attacker
will be updated according to the reward.
Reward_d, Next Observation (the less points disconnected, the larger the reward is)
Reward_a, Next Observation (the less points disconnected, the smaller the reward is)
Environment

(a) Observation and Action

(b) Reward and Next Observation

Figure 3: Framework

We proposed a framework composed of three parts: defender, attacker and environment as shown in Figure 3.

At the beginning of each time step, the environment generates a random directed graph and sends its adjacent matrix to defender and attacker as their observations (we use this terminology observation to make the framework compatible with Reinforcement Learning model).

Next, the defender and attacker will tell the environment their actions (which edges to defend/attack, use this terminology for compatibility, too).

And then, we enter the next time step. The environment will analyze their actions and give them corresponding rewards, and the observations of the next step. The rewards can have several designs, but it has to meet the requirement that the more vertices lose connection with the source point, the less rewards the defender will get while the more rewards the attacker will get. We define the reward that attacker and defender will get as:

$$r^{\text{attacker}} = \frac{\#\text{vertices lose connection with the source point}}{|V| - 1}$$

$$r^{\text{defender}} = -r^{\text{attacker}}$$

3.2 Model

We provide three models, an RL-based model, a stochastic regime model and a rule-based model.

3.2.1 RL-based Model

We have to define the *Observation*, *Action*, *Reward* of the Reinforcement Learning process.

Observation

[4] This section is written by H. Guo and J. Chen

The observation is the randomly generated directed graph G, as a flattened adjacent matrix.

Action

The classic Reinforcement Learning Algorithms only choose one action in the action space at a time step. If we want to use them directly, we need to treat every combination of K possible edges as one kind of action, which will cause the action space to grow exponentially with the number of edges. However, we made some modification, we set choosing every edge as each action. So the action space becomes $\{1, 2, \cdots, |E|\}$. Actor Critic algorithm will compute the probability of choosing each action. We choose K edges with the largest probability as the final action.

Reward

The definition of reward is equal to the reward in the framework.

Update

We have to make modification to the update procedure because we have changed the definition of the action.

The original policy update equation is:

$$\nabla_\theta \log \pi(a_t|s_t, \theta) A(s_t, a_t; \theta)$$

We modify it into the following equation:

$$\frac{1}{K} \sum_{k=1}^{K} \nabla_\theta \log \pi(a_t^k|s_t, \theta) A(s_t, a_t^k; \theta)$$

3.2.2 Stochastic-Regime Model

We implement a Stochastic-Regime Model as a baseline to make a comparison with the RL-based model. The model adopts a simple strategy, where both the attacker and the defender randomly select edges to delete or protect. In the experiments part, we think the RL based model is effective just if its attacker performs high success rate of reducing the connectivity when the defender is based on stochastic strategy, and its defender performs high success rate of protecting the connectivity when the attacker is based on stochastic strategy. The real performance of the stochastic-regime model can be seen in section 4.

3.2.3 Rule-based Model

We also implement a rule-based model, in which the attacker and the defender follow certain rules to select edges. There are many ways to do so, like selecting edges randomly, selecting cutting edges preferentially, selecting edges that are not in the strongly connected components of the graph, etc. In reality, a network is always complex and has a large amount of edges, and an attacker's ability might be limited, that is to say, K should be far less than $|E|$. Therefore, we give priority to the edges adjacent to the source point. This strategy is quite efficient for the attacker to destroy the connectivity against a defender based on stochastic strategy. However, as we can see in section 4, when the rule-based attacker face the same rule-based defender, its success rate reduce greatly.

4 Experiments[5]

In this part, we introduce the experiments we carried. Note that in the following experiments, the network has 5 vertices, 16 edges. Attacker/defender chooses 3 edges to defend/attack. And by 'baseline model', we mean model adopting rule-based defend/attack regime.

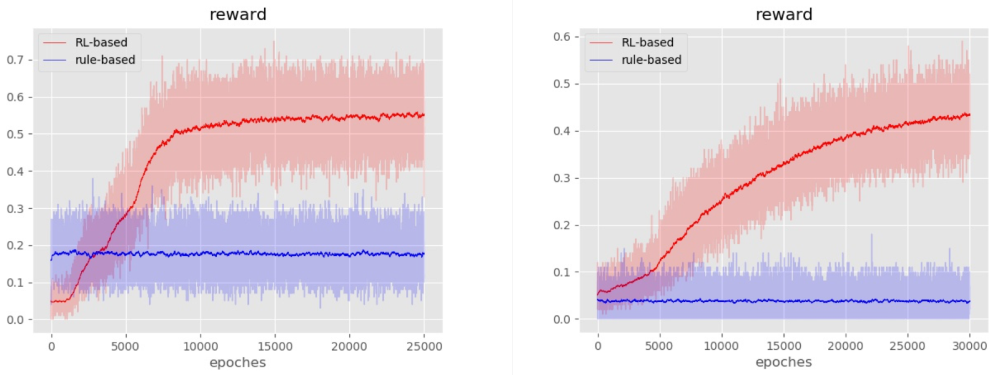
4.1 RL-based Attacker v.s. Stochastic-Regime Defender

First, we train a RL-based attacker model against a defender adopting stochastic defending regime. The results are as in 4(a), where the red line represents the rewards that our RL-based attacker gets, and the blue line represents the rewards the baseline attacker gets. It's easy to notice our RL-based attacker outperforms the baseline after less than $5,000$ epochs.

[5] This section is written by H. Guo

4.2 RL-based Attacker v.s. Rule-based Defender

Then, we train the same RL-based attacker. However, the defender this time is a rule-based model. The results are as in 4(b). Rule-based model is slightly stronger than the stochastic model. So we notice that the improvement of the RL-based attacker's performance becomes slower than in the last experiment. But near the end of the training, the reward is very close to that in the last experiment. In contrast, the rule-based attacker becomes useless facing the defender with the same rule.



(a) RL-based attacker v.s. stochastic-regime defender

(b) RL-based attacker v.s. rule-based defender

4.3 RL-based Defender v.s. Pre-trained RL-based Attacker

Now, we train an RL defender model against the RL-based attacker model already trained in 4.1 (free the parameter). As we can see in 4, our RL-based model outperforms baseline and can defend almost every attack of the smart trained attacker.

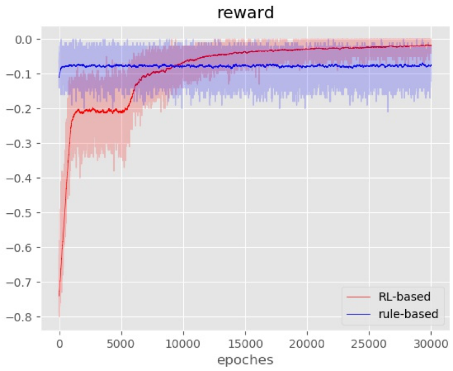


Figure 4: RL-based defender v.s. pre-trained RL-based attacker

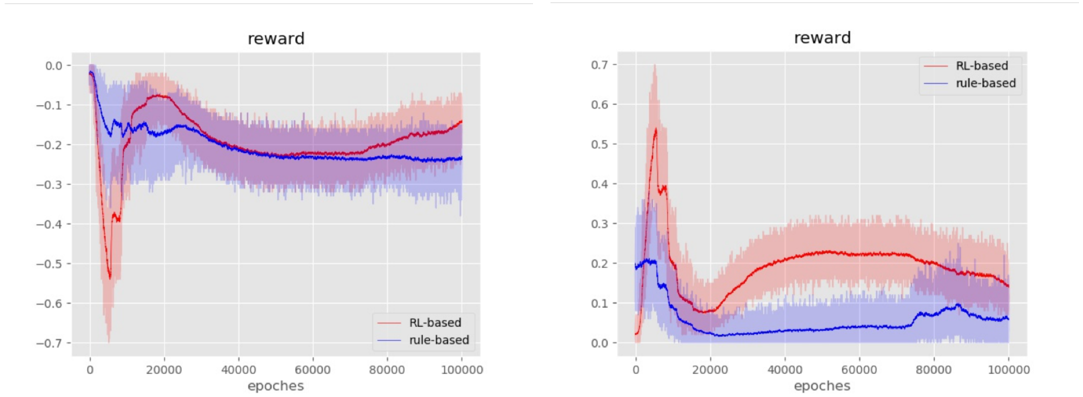
4.4 RL-based Defender v.s. RL-based Attacker

In the last experiment, we mimic the training of GAN, training an RL-based attacker and an RL-based defender at the same time. In Fig.5, we can see that both defender and attacker outperform the baselines in the end, indicating that the GAN-style training is effective.

5 Conclusion[6]

Our efforts are as follows:

[6] This section is written by H. Guo



(a) RL-based defender

(b) RL-based attacker

Figure 5: RL-based defender v.s. RL-based attacker

1. We surveyed about ten papers related to secure connectivity under the risk of adversarial model, and chose to focus on protecting the connectivity against smart attacker.
2. We proposed an adversarial learning framework composed of a defender, an attacker and an environment to address the above issue.
3. We also provided three different attacker/defender models for framework, including an Reinforcement-Learning-based model.
4. We did a lot of experiments on our framework, and the result shows that our proposed framework and models perform very well.

Our division of labour is shown in Table 1.

Name	Survey	RL-based Model	Baseline Model	Experiments	Report
Hongyi Guo	✓	✓	✓	✓	✓
Jikun Chen	✓		✓		✓

Table 1: Division of Labour

References

[1] Pan Zhou, Lin Cheng, and Dapeng Oliver Wu. Near optimal adaptive shortest path routing with stochastic links states under adversarial attack. *CoRR*, abs/1610.03348, 2016.

[2] Gergely Neu, András György, and Csaba Szepesvári. The adversarial stochastic shortest path problem with unknown transition probabilities. In Neil D. Lawrence and Mark A. Girolami, editors, *AISTATS*, volume 22 of *JMLR Proceedings*, pages 805–813. JMLR.org, 2012.

[3] Paul Balister, Stefanie Gerke, and Andrew McDowell. Adversarial resilience of matchings in bipartite random graphs. *J. Combin*, 8:79–92, 2017.

[4] Abraham D Flaxman, Alan M Frieze, and Juan Vera. Adversarial deletion in a scale-free random graph process. *Combinatorics, Probability and Computing*, 16(2):261–270, 2007.

[5] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, February 2015.

[6] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International Conference on Machine Learning*, pages 1928–1937, 2016.

[7] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.