mathematics

MDPI

Article

Playing Repeated Stochastic Security Games Against Non-Stationary Attackers

Ling Chen [1,*] and Runfa Zhang [2,3]

[1] School of Mathematics and Statistics, Taiyuan Normal University, Jinzhong 030619, China
[2] School of Automation and Software Engineering, Shanxi University, Taiyuan 030013, China; zrf@sxu.edu.cn
[3] Hubei Key Laboratory of Applied Mathematics, Hubei University, Wuhan 430062, China
* Correspondence: chenlingmath@163.com

Abstract

This paper investigates a repeated stochastic security game against a non-stationary attacker. Most of the work to date assumes that the defender has a repeated interaction with a fixed type of attacker. In fact, the defender is more likely to encounter changing attackers in multi-round games. A defender faces an attacker whose identity is unknown. The attacker type changes stochastically over time and the defender cannot detect when these changes occur. We adopt the BPR (Bayesian Policy Reuse) algorithm to detect the switches of the attacker, and the defender could play the accurate policy correspondingly. The experiment results show that BPR algorithm could accurately detect switches and help the defender gain more utilities than the EXP3-S algorithm.

Keywords: repeated stochastic security games; non-stationary environments; switching strategies; Bayesian Policy Reuse

MSC: 91-10

1. Introduction

In the past decades, Stackelberg security games have been used extensively in real-world systems, such as to protect wildlife, ports, and airports. Considering that the defender and the attacker interact multiple times in the real world, Refs. [1–3] extend the Stackelberg game model to a repeated one. In previous repeated security games, the defender protects a set of targets from being attacked by one attacker type. In each round, the attacker best responds to the deployment of defender or adopts a combination of the BR strategy and a fixed stubborn one [4]. However, the attacker would not consistently respond to the defender with a same behavior. They may change their behavior during the game. Refs. [5,6] investigate repeated security games with unknown (to the defender) game payoffs and attacker behaviors. They propose an efficient defender strategy based on an adversarial online learning framework. Refs. [7,8] study a repeated Stackelberg security game model, which assumes that the attackers are not all the same. In each round, the defender commits to a mixed strategy based on the history so far, and an adversarially chosen attacker from a set best responds to that strategy. Ref. [9] studies the equilibrium solution in discounted stochastic security games. It is a pity that these studies fail to consider a scenario where the attacker will change their behavior depending on the context they encounter. In fact, in a repeated game, the attacker may select an action that does not actually yield the best immediate reward to avoid revealing sensitive private

check for updates

Academic Editor: Antanas Cenys

Received: 15 July 2025
Revised: 16 August 2025
Accepted: 19 August 2025
Published: 22 August 2025

Citation: Chen, L.; Zhang, R. Playing Repeated Stochastic Security Games Against Non-Stationary Attackers. *Mathematics* **2025**, *13*, 2697. https://doi.org/10.3390/math13172697

Copyright: © 2025 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

information [10]. The attacker may change their behavior from a fully rational behavior, i.e., a best-response strategy, to a bounded rational behavior, such as Quantal Response (QR) behavior [11,12]. And they did not study how to detect these changes in repeated security games. This paper will consider a repeated stochastic security game where the defender interacts with a non-stationary attacker. The attacker may best respond to the defender in the beginning. And they may change their way of behavior in a round to a fully adversarial one. After several rounds, they may change their behavior to another one. Those changes are unknown to the defender. The defender's task is to learn the opponent type and detect those changes in time in a non-stationary environment. Related work in repeated stochastic games has proposed different learning algorithms [13–18]. However, they are not able to deal with unannounced changes when facing different opponent types. Bayesian and type-based approaches are a natural fit for our game setting. Bayesian Policy Reuse (BPR) [19] is an algorithm for efficiently responding to a novel task instance, assuming the availability of a policy library and prior knowledge of the performance of the library over different tasks. Based on observed 'signals', which are correlated with policy performance, it reuses a policy from the policy library. Similarly, the defender in this paper is faced with an attacker whose behavior may change in a type library. The type library includes five behavior types [5]. For each type of attacker behavior, we could obtain the defense policy in advance. Thus the BPR algorithm is adopted in our repeated stochastic game. Our main contributions are as follows: (1) providing five finite-time general MINLPs (mixed-integer nonlinear programs) for computing defender's strategies, (2) adopting the BPR algorithm to detect the changes of attacker behavior in time, and (3) comparing with the EXPS-3 algorithm to demonstrate that the BPR algorithm exhibits superior performance in enhancing the defender's utility in our game setting.

The structure of this article is as follows: The repeated stochastic game setting is presented in Section 2. The attacker behavior types and corresponding defense policies are given in Section 3. Section 4 adopts the BPR algorithm to solve our problem based on the results of Section 3. The experiment results are given in Section 5. Conclusions are provided in Section 6.

2. Game Setting

Unlike classical repeated Stackelberg games that model one defender playing against one attacker in multiple rounds, we study a repeated stochastic security game where the defender faces a non-stationary attacker. The attacker may select an action that does not actually yield the best immediate reward to avoid revealing sensitive private information to the defender. They may change their policy (i.e., the action choice rule) during repeated interaction. A defender faces an attacker whose identity is unknown to them. The attacker type changes stochastically over time and the defender cannot detect when these changes occur.

The repeated stochastic security games in a 2D grid can be represented as a tuple $< P, S, A, T, \Pi, R >$, with the following items:

- $P = \{D, A\}$: This is the set of players, where D denotes the defender and A denotes the attacker.
- S: The state space is the node set in the grid plus a special "absorbing" state; the game enters this state when the attacker attacks and remains there forever.
- $A = A_D \times A_F$: This is the set of possible action pairs, where $A_D = \{Up, Down, Left, Right\}$ is the set of actions of the defender and A_F specifies whether to attack one target in the grid or wait.
- T is a transition function. $T_{ss'}^{a_D, a_F}$ is the probability that state s transits into state s' after playing action $a_D \in A_D, a_F \in A_F$.

- $\Pi = (\Pi_D, \Pi_F)$ is the set of policy pairs, where $\Pi_D : S \to A_D$ is the policy set of the defender and $\Pi_F : S \to A_F$ is the policy of the attacker.
- $R = (R_D, R_F)$ is the pair of payoff functions $R_D, R_F : S \times A \to R$.

Let U_D and U_F denote the utility functions for the defender and attacker, respectively. For arbitrary policies $\pi_D \in \Pi_D$ and $\pi_F \in \Pi_F$,

$$U_D(s, \pi_D, \pi_F) = E[\sum_{t=1}^{\infty} \gamma_D^{t-1} R_D(s(t), \pi_D(h(t)), \pi_F(h(t)))|s(1) = s], \quad (1)$$

where the expectation is over the stochastic environment states, γ_D is a discount factor, and

$$R_D(s(t), \pi_D(h(t), \pi_F(h(t))) = \sum_{a_D \in A_D} \sum_{a_F \in A_F} \pi_D(a_D|h(t)) \pi_F(a_F|h(t)) R_D(s, a_D, a_F). \quad (2)$$

In Figure 1 we show an example of a small grid world.

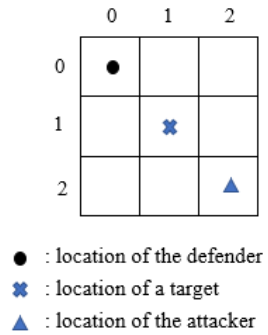


Figure 1. A small grid example.

In state t, the state s_t is represented as $[(0,0),(2,2)]$. If the defender chooses to go "Down" and the attacker chooses to go "Left", the state will become $[(0,1),(1,2)]$. A game will end when it enters an "absorbing" state. The "absorbing" state will be achieved if the attacker successfully attacks the target or the defender and the attacker are located in a same grid. In repeated stochastic security games, we consider five different attacker types, which together represent the majority of typical attacking models [5]. The attacker types are as follows:

- Uniform: An attacker with a uniformly mixed strategy;
- Adversarial: The attacker assumes that the defender will make the most unfavorable response to him and he will take a strategy to optimize the worst case. An adversarial attacker will play a maximin mixed strategy;
- Stackelberg: The attacker plays an optimal pure strategy according to Strong Stackelberg Equilibrium;
- Best Response: The attacker best responds to the defender mixed strategy in history;
- Quantal Response: The attacker responds to the defender mixed strategy by a QR model.

In a repeated stochastic security game, we denote the attacker types set by Γ. At each episode $t \in \{1, \cdots, T\}$ a process draws an attacker type $\gamma \in \Gamma$ from the set Γ to play a finite stochastic security game that yields a reward for a defender and attacker type γ. After the stochastic game terminates, the subsequent interaction commences. The attacker's type is redrawn for interactions over several repeated security games. The attacker may change their type randomly and the defender has an unknown distribution over the set Γ. The main purpose of the defender is to learn the attacker's behavior in a short time and to play an

optimal policy against it. Furthermore, it requires that the defender should detect fast that the attacker's type has changed into a different one and adjust their policy accordingly.

Under the stochastic security game framework, the attacker behavior is modeled through an MDP. For the Stackelberg attacker type and best-response attacker type, the strategy is pure. For the uniform attacker type, adversarial attacker type, and quantal response attacker type, the optimal strategy is mixed.

3. Bayesian Policy Reuse for Repeated Stochastic Security Game

In Section 2, we obtain that the defender should detect fast that the attacker's type has changed into a different one and adjust their policy accordingly. Bayesian and type-based approaches are a natural fit for our game setting. Considering the players as different agents, we adopt the Bayesian Policy Reuse (BPR) algorithm to solve our problem.

Bayesian Policy Reuse (BPR) was proposed as a framework to quickly determine the best policy to select when faced with an unknown task [20]. A task is defined as an MDP, $\mathcal{M} = \{S, A, T, R\}$, where S is the state set, A is the action set, T is the transition function, and R is the reward function. A policy specifies an action a for each state s. The accumulated reward can be computed as $U^{\pi} = \sum_{t=0}^{T} R_t$, where T is the length of the interaction and R_t is the immediate reward at stage t. We need to find an optimal policy $\pi^* = argmaxU^{\pi}$ for the defender by solving an MDP $\mathcal{M}$. We denote the previously solved task set by Γ. When faced with an unknown task $\hat{\gamma}$, BPR computes a probability distribution β over the set Γ to measure the degree to which $\hat{\gamma}$ matches the previously solved tasks based on the received signal. The belief is initialized with a prior probability. A signal is related to the performance of a policy, such as immediate rewards and episodic returns. The performance model $P(U|\gamma, \pi)$ is a probability distribution over the utility of a policy π on a task γ. Based on the performance model, the belief β at state t can be updated according to Bayes' rule:

$$\beta^t(\gamma) = \frac{P(\sigma^t|\gamma, \pi^t)\beta^{t-1}(\gamma)}{\sum_{\gamma' \in \Gamma} P(\sigma^t|\gamma', \pi^t)\beta^{t-1}(\gamma')} \tag{3}$$

In fact, playing against one attacker type for the defender is equal to solving one MDP in a stochastic security game. The decision-making process of attackers can be modeled as an MDP. The tasks in BPR correspond to the attacker types and the policies correspond to the optimal policies against those attacker types.

We should first obtain the policy library Π. To compute the optimal policy π_D^* of the defender against different attacker types, we build the following mixed-integer non-linear programmings (MINLPs).

Let $V_D(s)$ and $V_F(s)$ denote the expected utility function of the defender and attacker starting in state s. In state s, the defender plays a policy π_D and the attacker plays a policy π_F. The attacker's expected utility can be represented as

$$R_F(s, \pi_D, \pi_F) = \sum_{a_D \in A_D} \sum_{a_F \in A_F} \pi_D(a_D|s)\pi_F(a_F|s) \left(R_F(s, a_D, a_F) + \gamma_F \sum_{s' \in S} T_{ss'}^{a_D a_F} V_F(s') \right). \tag{4}$$

where γ_F is a discount factor.

Based on the above definitions, the defender's policies against the five attacker types can be obtained by building the following programmings.

The uniform attacker type has little dependence on the history and plays a uniformly random mixed strategy in each round. To defend against this kind of attacker type, the optimal policy of the defender can be computed by the programming (P1).

$$\max_{\pi_D,\pi_F} \sum_{s\in S} \alpha(s) R_D(s,\pi_D,\pi_F)$$

s.t.

$$\pi_F(a_F|s) \in [0,1], \forall s \tag{5}$$

$$\sum_{a_F\in A_F} \pi_F(a_F|s) = 1 \tag{6}$$

$$\pi_D(a_D|s) \geq 0, \forall s, \forall a_D \in A_D \tag{7}$$

$$\sum_{a_D\in A_D} \pi_D(a_D|s) = 1 \tag{8}$$

The objective is to maximize the expected utility of the defender with respect to the distribution of initial states. Constraints (5) and (6) compute the mixed strategy of the attacker. Constraints (7) and (8) decide the mixed strategy of the defender.

To defend against an adversarial attacker type, we can build the programming (P2).

$$\max_{\pi_D,\pi_F,V_D} \sum_{s\in S} \alpha(s) V_D(s)$$

s.t.

$$\pi_F(a_F|s) \in [0,1], \forall s \tag{9}$$

$$\sum_{a_F\in A_F} \pi(a_F|s) = 1 \tag{10}$$

$$\pi_D(a_D|s) \geq 0, \forall s, \forall a_D \in A_D \tag{11}$$

$$\sum_{a_D\in A_D} \pi_D(a_D|s) = 1, \tag{12}$$

$$R_D(s,\pi_D,\pi_F) \geq V_D(s) \tag{13}$$

Constraints (9) and (10) are the mixed strategy of the attacker. Constraints (11) and (12) are the mixed strategy of the defender. Constraint (13) corresponds to a maxmin strategy, because an adversarial attacker only cares about minimizing the defender's utility. The adversarial attacker plays a policy π_F to minimize the defender's utility $V_D(s)$, while the defender would try their best to maximize the utility.

A Stackelberg attacker type requires that the attacker plays a best-response strategy and breaks ties in favor of the defender. We can compute a Strong Stackelberg Equilibrium (SSE) to find the optimal policies of the defender and Stackelberg attacker type. The SSE solution can be computed by the following programming (P3).

$$\max_{\pi_D,\pi_F,V_D,V_F} \sum_{s\in S} \alpha(s) V_D(s)$$

$$\text{s.t.} \pi_F(a_F|s) \in \{0,1\}, \forall s \in S \tag{14}$$

$$\sum_{a_F\in A_F} \pi_F(a_F|s) = 1 \tag{15}$$

$$\pi_D(a_D|s) \geq 0 \tag{16}$$

$$\sum_{a_D\in A_D} \pi_D(a_D|s) = 1 \tag{17}$$

$$0 \leq V_F(s) - R_F(s,\pi_D,a_F) \leq (1-\pi_F(a_F|s)) \cdot Z, \forall s \in S \tag{18}$$

$$V_D(s) - R_D(s,\pi,a_F) \leq (1-\pi_F(a_F|s)) \cdot Z, \forall s \in S \tag{19}$$

Constraints (18) are used to compute the attacker's best response π_F to a defender's policy π_D. The first inequality represents the requirement that the attacker value $V_F(s)$

in state s maximizes their expected utility over all possible choices a_F they can make in this state. The second inequality ensures that if the attacker chooses an action a_F in state s, i.e., $\pi_F(a_F|s) = 1$, $V_F(s)$ exactly equals the attacker's expected utility in that state. If an action a_F is not chosen in state s, i.e., $\pi_F(a_F|s) = 0$, the right-hand-side is a large constant and the inequality has no force. Based on the best response of the attacker, constraints (19) are used to compute the defender's expected utility. When the attacker chooses a_F and $\pi_F(a_F|s) = 1$, the defender's utility must equal the expected utility when the attacker plays a_F. Otherwise the inequality has no force.

To defend against a best-response attacker type, we can build the programming (P4).

$$\max_{\pi_D, \pi_F} \sum_{s \in S} \alpha(s) V_D(s)$$

$$\text{s.t.} \pi_F(a_F|s) \in \{0,1\}, \forall s \in S \tag{20}$$

$$\sum_{a_F \in A_F} \pi_F(a_F|s) = 1 \tag{21}$$

$$\pi_D(a_D|s) \geq 0 \tag{22}$$

$$\sum_{a_D \in A_D} \pi_D(a_D|s) = 1 \tag{23}$$

$$0 \leq V_F(s) - R_F(s, \pi_D, \pi_F) \leq (1 - \pi_F(a_F|s)) \cdot Z, \forall s \in S \tag{24}$$

While the framework of programming (P4) maintains the same structure as (P3), it simplifies the model by only requiring a best-response strategy from the attacker, and the constraints on defender's policy could be deleted.

To defend against a Quantal Response attacker type, the optimal policy of the defender can be computed by solving programming (P5).

$$\max_{\pi_D, \pi_F} \sum_{s \in S} \alpha(s) R_D(s, \pi_D, \pi_F)$$

$$\text{s.t.} \pi_F(a_F|s) = \frac{e^{\lambda \hat{R}_F(s, \pi_D, a_F)}}{\sum_{a'_F \in A_F} e^{\lambda \hat{R}_F(s, \pi_D, a'_F)}}, \forall s \in S \tag{25}$$

$$\pi_F(a_F|s) \in [0,1] \tag{26}$$

$$\sum_{a_F \in A_F} \pi_F(a_F|s) = 1 \tag{27}$$

$$\pi_D(a_D|s) \geq 0 \tag{28}$$

$$\sum_{a_D \in A_D} \pi(a_D|s) = 1 \tag{29}$$

If the attacker is a Quantal Response type, they would choose a_F in state s according to the probability distribution in constraints (25). QR predicts a probability distribution over attacker actions where actions with higher utility have a greater chance of being chosen.

(P1)–(P5) are all mixed-integer nonlinear programmings.

In multi-round stochastic security games, the attacker may switch among the five behavior types. The main goal of the defender is to quickly detect the changes of the attacker's behavior and then respond with an accurate policy. We need to design an algorithm that computes a belief over the possible attacker types. The belief is updated at every interaction.

4. BPR-MINLP Algorithm

In this section, we propose a BPR-MINLP algorithm to detect the changes of attacker's behavior. The algorithm is composed of two phases. The first is an offline phase where each

MINLP is solved. We first assume the attacker type is observed by the defender. Through computing programmings (P1)–(P5), we could obtain the optimal policies of the defender against the five kinds of attacker types. The policy generation is given in Algorithm 1. The second phase is an online phase where a belief-based approach is used to detect the attacker's identity and act with the corresponding policy. The defender will receive an accumulated reward when a stochastic game finishes, which is a signal to update the belief. The belief is updated at every state in a stochastic game and at every stochastic security game. Algorithm 2 combines the results of MINLPs and updating rule in BPR algorithm to detect the changes of attacker.

Algorithm 1: Policy generation

1. $\Pi = \emptyset$
2. **for** each attacker type $\gamma \in \Gamma$ **do**
3. attacker type γ is announced
4. solving the MINLP to learn a policy π_{new} facing γ
5. $\Pi = \Pi \cup \pi_{new}$

Algorithm 2: BPR-MINLP

Input: Policy library Π, prior probabilities $P(\Gamma)$,
performance models $P(U|\Gamma, \Pi)$, episodes K.
1. Initialize beliefs $\beta^0(\Gamma) = P(\Gamma)$
2. **for** each episodes $k = 1, \cdots, K$ **do**
3. compute $v_\pi = V(\pi, \beta^{k-1})$ for all $\pi \in \Pi$
4. $\pi^k = argmax_{\pi \in \Pi} v_\pi$
5. Use π^k against the attacker type
6. Obtain observation signal σ^k from the interaction
7. Update belief β^k according to Equation (3)

5. Experiment

In this section, we evaluate our proposed algorithm on a stochastic game represented as a 3×3 grid and 10×10 grid. Taking the game in the 3×3 grid world as an example, the defender has up to four actions depending on which nodes there are in - left, right, up, and down. The attacker is to choose one target to attack or wait. We performed all experiments using the Python 3.10.10 solver on a 2.3 GHz Intel Core i7 with 8 GB memory.

We should first obtain the policy library of the defender against five attacker types through Algorithm 1. All programmings are nonlinear. We adopt the Gurobi solver to solve these programmings. The utility parameters of players are drawn from $[-100,100]$. The discount factors $\gamma_D = \gamma_F = 0.95$. The results are stored in the policy library.

We run the Algorithm 2 in a 200-round security game. Figures 2 and 3 show the belief evolution in a 3×3 grid world and 10×10 grid. In Figure 2, the beliefs of (P3) and (P4) are at a lower level and those of (P1) and (P5) are at a higher level. This means that the BPR-MINLP algorithm believes that the attacker type is (P1) or (P5) with a higher probability. In (P1) and (P5), the attacker strategies are all mixed. In a 3×3 grid with only nine states, these two types are similar. When the attacker behavior changes into another one in stage 125, (P2) becomes the dominant type. In Figure 3, we set the negative value of the defender's reward as a punishment for the attacker. The maximin strategy is the same as the SSE strategy. The trajectories of (P2) and (P3) coincide. When the attacker type changes from (P5) to (P4), we can see that the algorithm can also detect the changes and identify the true attacker type.

The two solid lines in Figure 4 show the cumulative rewards in a 3×3 grid and 10×10 grid, respectively. We can see that the cumulative reward in the 10×10 grid increases faster than that of a game in 3×3. The state set of the 10×10 grid is much larger and the Algorithm 2 could better perform in a more complex environment. To verify the effectiveness of the BPR algorithm in our game setting, we compare it with the EXP3-S algorithm [21]. The cumulative rewards using the EXP3-S algorithm in a 3×3 grid world and 10×10 grid world are represented by the two dashed lines in Figure 4. We can see that the EXP3-S algorithm in the 3×3 grid world has poor performance and the cumulative reward is nearly -2500. The EXP3-S algorithm in the 10×10 grid world has the worst performance and the cumulative reward is smaller than $-20{,}000$. The results show that the Algorithm 2 outperforms the EXP3-S algorithm in improving the defender's utility. The belief update mechanism of the BPR algorithm could adapt to a changing environment and accurately detect the changes of the attacker's behavior. The EXP3-S algorithm makes wrong decisions in a complex environment.

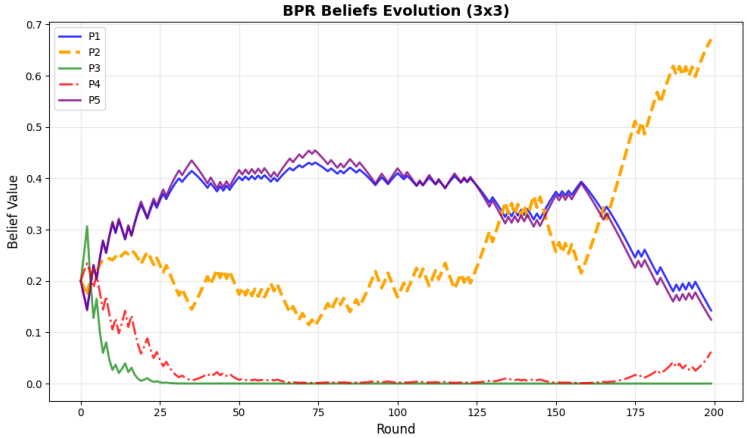


Figure 2. Belief values in a 3×3 grid world.

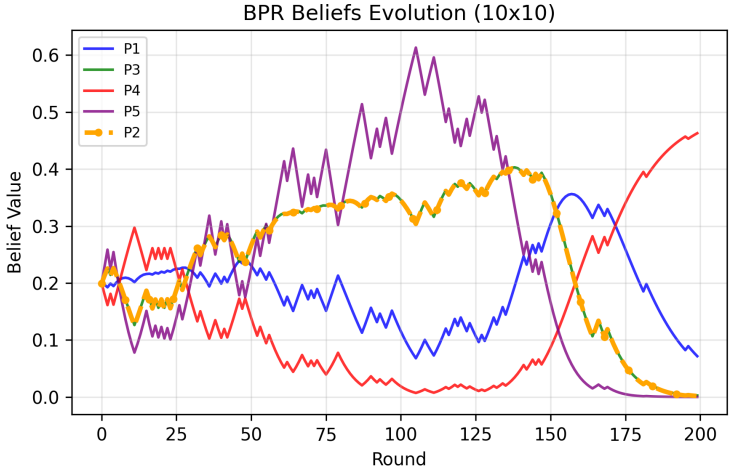


Figure 3. Belief values in a 10×10 grid world.

Figures 5–7 show the comparison results of the system performance in a 3×3 grid and 10×10 grid. We can see that the runtime after normalizing 3×3 is nearly 0.05 and nearly 1.0 for 10×10. This fits our expectation that the complexity is proportional to the running time. In Figure 6, we can see that the memory usage of 3×3 remains stable at 68.2. And the value of 10×10 remains stable at 69.4. The games in the two grid worlds all converge within 200 rounds. This shows that the algorithm has good convergence and is not significantly affected by grid size.

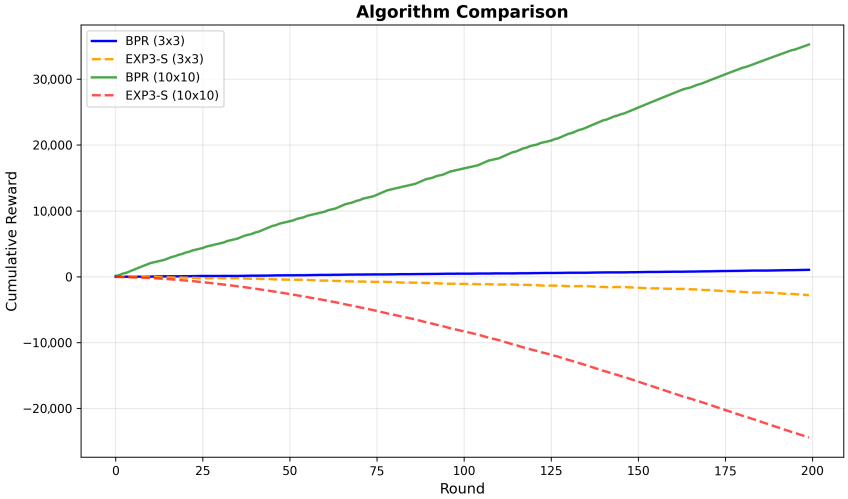


Figure 4. The cumulative reward using the BPR algorithm and the comparison results with the EXP3-S algorithm.

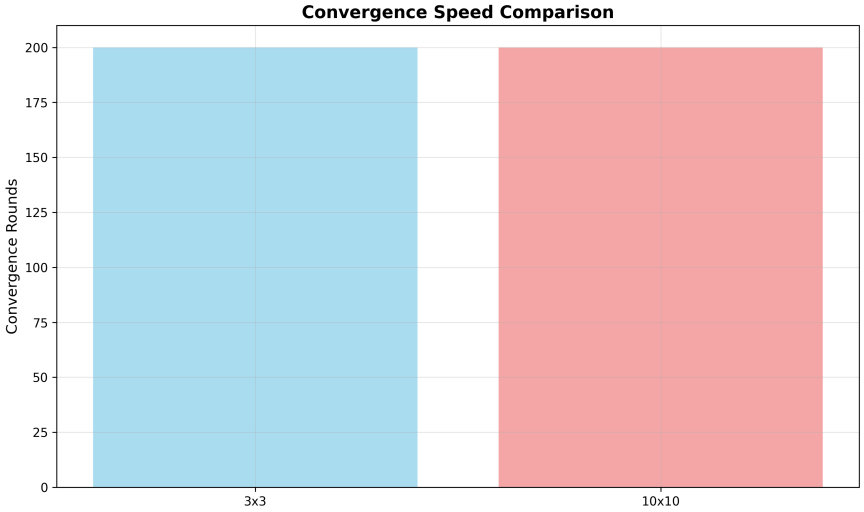


Figure 5. Convergence speed.

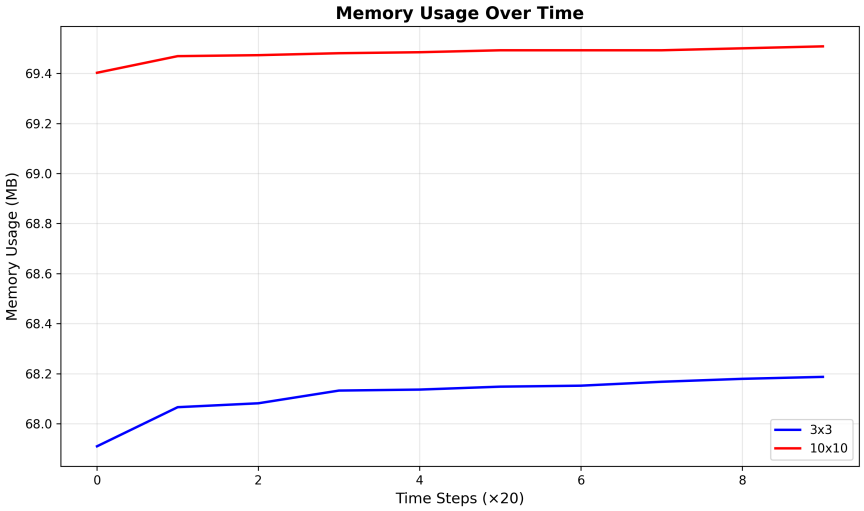


Figure 6. Memory usage over time.

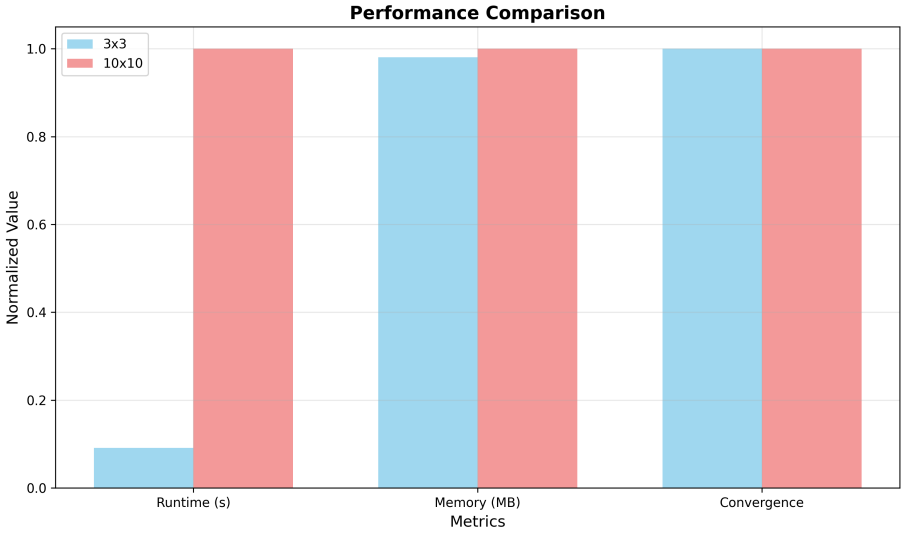


Figure 7. Performance comparison results of 3×3 grid world and 10×10 grid world.

Figures 8 and 9 show the sensitivity analysis results of parameters π and λ in the QR behavior model. In Figure 8, we can see that the cumulative reward increases with the increase in π. This means that a higher belief parameter helps the algorithm better adapt to the environmental change. In our game setting, it is better that the value of π should be set within the range of 0.85 to 0.95. Figure 9 shows the trend of cumulative reward with the variation in parameter λ. Parameter λ quantifies the degree of rationality, where higher values indicate greater levels of rational decision-making. When λ increases from 0 to 1.5, we can see that the cumulative reward of the defender decreases. λ in this range means the attacker has a lower level of rationality. There is a significant degree of blindness when reasoning their strategy. The defender may highly utilize known strategies and be trapped in an inefficient equilibrium solution. If λ is not less than 1.5, the behavior of attacker with higher rationality is much easier to infer. The defender's cumulative reward will increase.

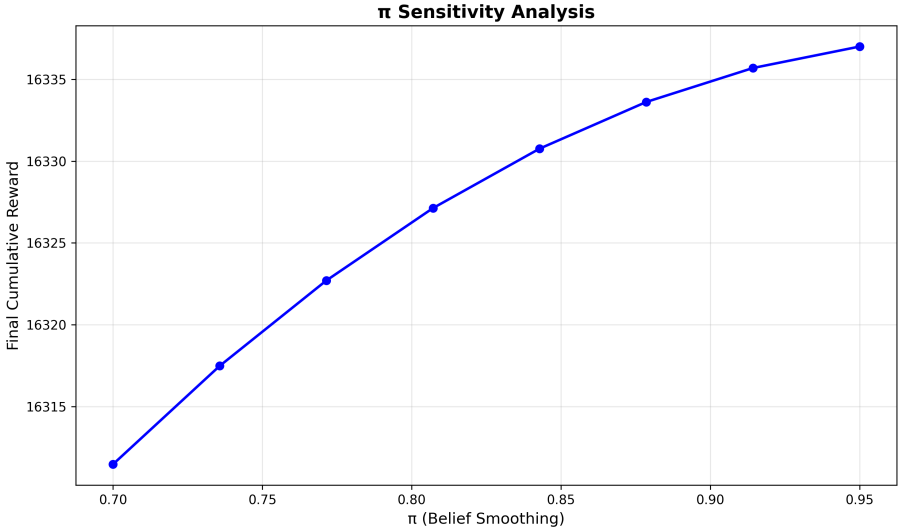


Figure 8. Cumulative rewards under different values of π.

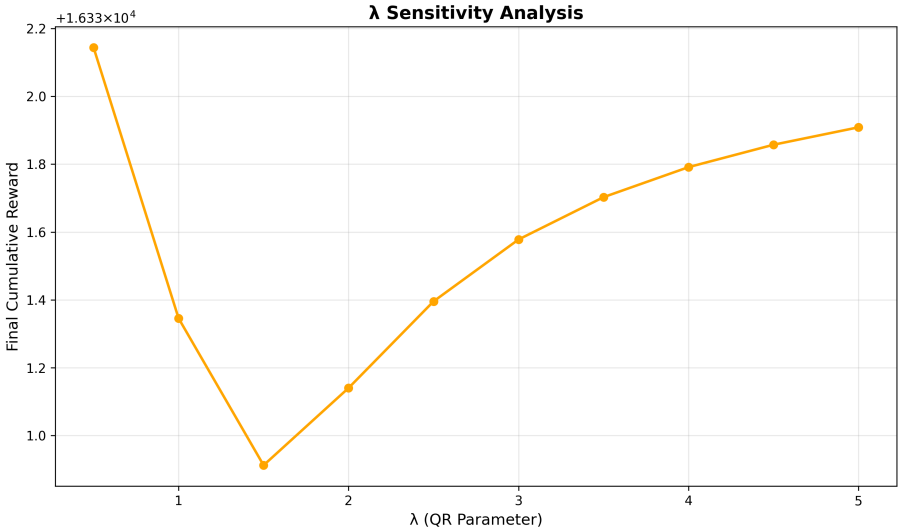


Figure 9. Cumulative rewards under different values of λ.

Figure 10 is the ROC curve for change detection. We can see that the true positive rate increases rapidly and the value of AUC is 0.941. This shows that Algorithm 2 could achieve a high true positive rate at a low false positive rate. This proves that the algorithm could detect the changes of attacker type in time.

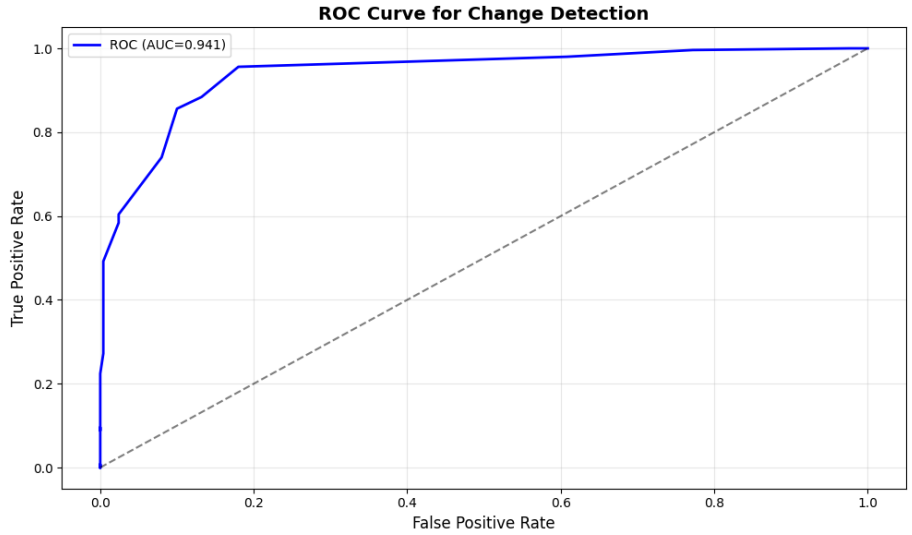


Figure 10. ROC curve for change detection.

6. Conclusions

In this paper, we investigate a repeated stochastic security game against a non-stationary attacker. The defender faces an attacker whose identity is unknown to them and where every few interactions a random process selects a new attacker type from the population, while the defender does not know when these changes happen. To detect the changes promptly, we adopt the BPR algorithm to solve our problem. The policy library against each kind of attacker type is obtained by building an MINLP. Compared with other online learning algorithms, such as the EXP3-S algorithm, the experiment results show that our proposed algorithm could detect these changes correctly and outperform in increasing the utility of the defender. In future work, we will explore more attacker behavior types, such as atheory of mind player, and consider a more realistic repeated security game.

Author Contributions: Conceptualization, L.C.; Methodology, L.C.; Formal analysis, R.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This work is supported by the Tianyuan Fund for Mathematics of the National Natural Science Foundation of China (Grant No. 12426105), the Fundamental Research Program of Shanxi Province (No. 202203021222251, 202403021222001), and the Scientific and Technological Innovation Programs (STIP) of Higher Education Institutions in Shanxi (No. 2024L022) and funded by the Open Foundation of Hubei Key Laboratory of Applied Mathematics (Hubei University) under No. HBAM202401 and the "Wen Ying Young Scholars" Talent Project of Shanxi University (No. 138541088).

Data Availability Statement: The original contributions presented in this study are included in the article. Further inquiries can be directed to the corresponding author.

Conflicts of Interest: The author declares no conflicts of interest.

References

1. Kar, D.; Fang, F.; Delle Fave, F.; Sintov, N.; Tambe, M. "A Game of Thrones" When Human Behavior Models Compete in Repeated Stackelberg Security Games. In Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems, Istanbul, Turkey, 4–8 May 2015; ACM: New York, NY, USA, pp. 1381–1390.
2. Wang, B.; Zhang, Y.; Zhou, Z.H.; Zhong, S. On repeated stackelberg security game with the cooperative human behavior model for wildlife protection. *Appl. Intell.* **2019**, *49*, 1002–1015. [CrossRef]
3. Alcantara-Jiménez, G.; Clempner, J.B. Repeated Stackelberg security games: Learning with incomplete state information. *Reliab. Eng. Syst. Saf.* **2020**, *195*, 106695. [CrossRef]
4. Cheng, Z.; Chen, G.; Hong, Y. Zero-determinant strategy in stochastic Stackelberg asymmetric security game. *Sci. Rep.* **2023**, *13*, 11308. [CrossRef] [PubMed]
5. Xu, H.; Tran-Thanh, L.; Jennings, N.R. Playing repeated security games with no prior knowledge. In Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems, Singapore, 9–13 May 2016; ACM: New York, NY, USA, 2016; pp. 104–112.
6. Garnaev, A.; Baykal-Gursoy, M.; Poor, H.V. Security games with unknown adversarial strategies. *IEEE Trans. Cybern.* **2015**, *46*, 2291–2299. [CrossRef] [PubMed]
7. Balcan, M.F.; Blum, A.; Haghtalab, N.; Procaccia, A.D. Commitment without regrets: Online learning in stackelberg security games. In Proceedings of the Sixteenth ACM Conference on Economics and Computation, Portland, OR, USA, 15–19 June 2015; ACM: New York, NY, USA, 2015; pp. 61–78.
8. Hammar, K.; Stadler, R. Learning Near-Optimal Intrusion Responses Against Dynamic Attackers. *IEEE Trans. Netw. Serv. Manag.* **2024**, *21*, 1158–1177. [CrossRef]
9. Vorobeychik, Y.; Singh, S. Computing stackelberg equilibria in discounted stochastic games. In Proceedings of the 26th AAAI Conference on Artificial Intelligence, Toronto, ON, Canada, 22–26 July 2012; AAAI Press: Palo Alto, CA, USA, 2012; pp. 1478–1484.
10. Nguyen, T.H.; Wang, Y.; Sinha, A.; Wellman, M.P. Deception in finitely repeated security games. In Proceedings of the 33th AAAI Conference on Artificial Intelligence, Honolulu, HI, USA, 27 January–February 2019; AAAI Press: Palo Alto, CA, USA, 2012; pp. 2133–2140.
11. Mckelvey, R.D.; Palfrey, T.R. Quantal Response Equilibria for Normal Form Games. *Games Econ. Behav.* **1995**, *10*, 6–38. [CrossRef]
12. Yang, R.; Ordonez, F.; Tambe, M. Computing optimal strategy against quantal response in security games. In Proceedings of the 2012 International Conference on Autonomous Agents and Multiagent Systems, Valencia, Spain, 4–8 June 2012; ACM: New York, NY, USA, 2012; pp. 847–854.
13. Hernandez-Leal, P.; Kaisers, M. Learning against sequential opponents in repeated stochastic games. In Proceedings of the 3rd Multi-Disciplinary Conference on Reinforcement Learning and Decision Making, Ann Arbor, MI, USA, 11–14 June 2017; Volume 25, pp. 52–56.
14. Chen, H.; Liu, Q.; Fu, K.; Huang, J.; Wang, C.; Gong, J. Accurate policy detection and efficient knowledge reuse against multi-strategic opponents. *Knowl.-Based Syst.* **2022**, *242*, 108404. [CrossRef]
15. Hernandez-Leal, P.; Zhan, Y.; Taylor, M.E.; Sucar, L.E.; Munoz de Cote, E. Efficiently detecting switches against non-stationary opponents. *Auton. Agents Multi-Agent Syst.* **2017**, *31*, 767–789. [CrossRef]
16. Shen, Y.; Shepherd, C.; Ahmed, C.M.; Yu, S.; Li, T. Comparative DQN-Improved Algorithms for Stochastic Games-Based Automated Edge Intelligence-Enabled IoT Malware Spread-Suppression Strategies. *IEEE Internet Things J.* **2024**, *11*, 22550–22561. [CrossRef]

17. Bozkurt, A.K.; Wang, Y.; Zavlanos, M.M.; Pajic, M. Learning optimal strategies for temporal tasks in stochastic games. *IEEE Trans. Autom. Control* **2024**, *69*, 7387–7402. [CrossRef]
18. Hammar, K.; Li, T.; Stadler, R.; Zhu, Q. Adaptive security response strategies through conjectural online learning. *IEEE Trans. Inf. Forensics Secur.* **2025**, *20*, 4055–4070. [CrossRef]
19. Rosman, B.; Hawasly, M.; Ramamoorthy, S. Bayesian policy reuse. *Mach. Learn.* **2016**, *104*, 99–127. [CrossRef]
20. Hernandezleal, P.; Kaisers, M. Towards a Fast Detection of Opponents in Repeated Stochastic Games. In *Autonomous Agents and Multiagent Systems*; Springer: Cham, Switzerland, 2017; pp. 239–257.
21. Allesiardo, R.; Féraud, R. Exp3 with drift detection for the switching bandit problem. In Proceedings of the 2015 IEEE International Conference on Data Science and Advanced Analytics (DSAA), Paris, France, 19–21 October 2015; IEEE: New York, NY, USA, 2015; pp. 1–7.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.