

Czech Technical University in Prague
Faculty of Electrical Engineering
Department of Computer Science and Engineering



Master's Thesis

**Combining online learning and equilibrium computation
in security games**

by

Richard Klíma

Supervisor: Mgr. Viliam Lisý, M.Sc., Ph.D.

Master's programme: Open Informatics
Specialization: Artificial Intelligence

Prague, May 2015

Czech Technical University in Prague
Faculty of Electrical Engineering

Department of Computer Science and Engineering

DIPLOMA THESIS ASSIGNMENT

Student: **Richard Klíma**

Study programme: Open Informatics
Specialisation: Artificial Intelligence

Title of Diploma Thesis: **Combining online learning and equilibrium computation in security games**

Guidelines:

Game theory has been used to optimize physical security of airports, ports, or to improve protection of wildlife [1,2]. Game theoretic models require good approximations of players' options and motivations, which are often difficult and costly to obtain with a sufficient precision. Therefore, this thesis will investigate the tradeoffs between using imprecise game theoretic models and online learning of strategy from interactions with the opponents. The student will:

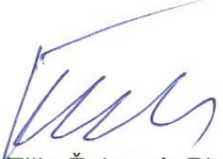
- review the existing literature on security games and online learning in adversarial setting;
- evaluate empirical performance of the game theoretic and online learning algorithms against adversary;
- design a combined method that starts with a strategy based on an imprecise game theoretic model and gradually improves the strategy based on interaction with the opponent; and
- study the applicability tradeoffs of each of these three approaches in practical applications.

Bibliography/Sources:


- [1] Tambe, Milind. Security and game theory: algorithms, deployed systems, lessons learned. Cambridge University Press, 2011.
- [2] An, Bo, David Kempe, Christopher Kiekintveld, Eric Shieh, Satinder Singh, Milind Tambe, and Yevgeniy Vorobeychik. "Security games with limited surveillance." Ann Arbor 1001 (2012): 48109.
- [3] Yang, Rong, Benjamin Ford, Milind Tambe, and Andrew Lemieux. "Adaptive resource allocation for wildlife protection against illegal poachers." In Proceedings of the 2014 international conference on Autonomous agents and multi-agent systems, pp. 453-460. International Foundation for Autonomous Agents and Multiagent Systems, 2014.
- [4] Bubeck, Sébastien, and Nicolo Cesa-Bianchi. "Regret Analysis of Stochastic and Nonstochastic Multi-armed Bandit Problems." Machine Learning 5, no. 1 (2012): 1-122.
- [5] Fudenberg, Drew. The theory of learning in games. MIT press, 1998.

Diploma Thesis Supervisor: Mgr. Viliam Lisý, MSc., Ph.D.

Valid until the end of the summer semester of academic year 2015/2016


doc. Ing. Filip Železný, Ph.D.
Head of Department




prof. Ing. Pavel Ripka, CSc.
Dean

Prague, March 26, 2015

Čestné prohlášení autora práce

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze dne 11. 5. 2015


.....
Podpis autora práce

Acknowledgement

I would like to thank my supervisor Mgr. Viliam Lisý, M.Sc., Ph.D. for his valuable advice and comments on this thesis. I would like to also thank Dr. Christopher Kiekintveld for a great cooperation on research work, which preceded the writing of this work. I need to mention that early framework of used game model in this work originally came from other students from the University of Texas at El Paso to whom I am also thankful. Last but not least I would like to greatly thank my parents for never-ending support.

Abstract

In the security domain there has recently been a big demand in developing effective approaches using game-theory tools. New algorithms are modeled in the framework known as Stackelberg security game. In this thesis, we study a two-player non-cooperative Stackelberg security game model inspired by application to border patrol. There are several known approaches to handle this game scenario. We firstly focus on online learning algorithms used in the well-known multi-armed bandit (MAB) problem. We investigate use of several online learning algorithms and their applications to security games. Then we focus on a game-theoretic approach by studying equilibria concepts. Both approaches, online learning algorithm and game-theoretic solution, have some limitations in real-world applications. The online learning algorithm has a very poor performance at the beginning of the game due to having no prior information about the opponent's strategy, while the game-theoretic strategy is not able to adapt to varying opponents. Therefore we present new combined algorithms, based on those two approaches, which address these limitations. We empirically test the proposed combined algorithms and show their advantages such as stability or adaptability under various game settings. The main contribution of the combined algorithms strategy is that it improves defender performance and thus is more effective in resource allocation in the proposed game model than previous approaches.

Keywords:

game theory, security games, online learning, Stackelberg game, Stackelberg equilibrium, Nash equilibrium, border patrol, multi-armed bandit problem.

Abstrakt

V současné době je v mnoha bezpečnostních odvětvích velká poptávka po vyvíjení efektivních přístupů, které používají teorii her. Nové algoritmy jsou modelovány v konceptu zvaném Stackelbergova bezpečnostní hra. V této práci se zabýváme Stackelbergovou bezpečnostní hrou se dvěma nespolutracujícími hráči inspirovanou problémem střežení hranic. Je známo několik přístupů, jak řešit takovýto herní model. Napřed se zaměřujeme na online učící algoritmy, které jsou používané v dobře známém problému mnohorukého bandity (multi-armed bandit problem - MAB). Prozkoumáme použití několika online učících algoritmů a jejich aplikaci do bezpečnostních her. Poté se zaměřujeme na herně teoretický přístup formou výpočtu herní rovnováhy. Oba přístupy online učících algoritmů a herně teoretického řešení mají v reálných aplikacích svá omezení. Online učící algoritmus má velmi špatný výsledek na začátku hry kvůli tomu, že nemá žádnou informaci o oponentově strategii. Herně-teoretické řešení se neumí přizpůsobit měnícímu se oponentovi. Tudíž navrhujeme kombinované algoritmy založené na těchto dvou přístupech, které řeší zmíněné omezení. Empiricky testujeme navržené algoritmy v několika herních nastaveních a ukážeme jejich výhody jako stabilita či schopnost přizpůsobit se. Tento hlavní přínos kombinovaných algoritmů vylepšuje výsledek obránce, a tudíž mu umožňuje efektivněji rozdělit své zdroje v navrženém herním modelu.

Klíčová slova

teorie her, bezpečnostní hry, online učení, Stackelbergova hra, Stackelbergova rovnováha, Nashova rovnováha, střežení hranic, multi-armed bandit problém.

Contents

List of Figures	xiv
List of Tables	xvi
1 Introduction	1
1.1 Related Work	2
1.2 Thesis Outline	5
2 Game Theory Introduction	7
2.1 Game Model	7
2.1.1 Normal-form game	7
2.2 Game Concepts	8
2.3 Stackelberg Security Games	9
2.4 Analyzed Game Model	9
2.5 Equilibrium in Game	11
2.5.1 Dominant strategy	11
2.5.2 Nash equilibrium	11
2.5.3 Stackelberg equilibrium	12
3 Learning in Games	15
3.1 Multi-armed Bandit Problem	15
3.1.1 Regret	16
3.1.2 Stochastic bandit problem model	16
3.1.3 Adversarial bandit problem model	17
3.2 Combinatorial Bandits	18
3.3 Convergence of Learning Algorithms	18
3.4 Learning Algorithms	18
3.4.1 Upper Confidence Bound (UCB) algorithm	19
3.4.2 EXP3 algorithm	19
3.4.3 Non-stationary UCB versions	21
3.4.3.1 Discounted UCB algorithm	21
3.4.3.2 Sliding-window UCB algorithm	22
3.4.4 Combinatorial EXP3 algorithm	23
3.4.4.1 Combinatorial sampling	23
3.5 Fictitious Play	23

4	Game Strategies	25
4.1	Player types	25
4.1.1	Defender types	25
4.1.2	Attacker types	26
4.2	Learning Algorithms Properties	27
4.2.1	UCB synchronization	27
4.2.2	Combinatorial EXP3 implementation	29
4.2.2.1	Projection heuristic	29
4.3	Game-theoretic Strategies	31
4.3.1	Stackelberg equilibrium strategy	31
4.3.2	Estimated Stackelberg equilibrium strategy	31
4.3.3	Motivating Stackelberg equilibrium strategy	32
4.4	Combined Algorithms	32
4.4.1	Combination algorithm 1	32
4.4.2	Combination algorithm 2	33
4.4.3	Combination algorithm 3	33
4.4.4	Combination algorithm 4	34
4.5	Combinatorial Combined Algorithms	34
4.5.1	Combinatorial COMB1	34
4.5.2	Combinatorial COMB2	35
4.5.3	Combinatorial COMB3	35
4.5.4	Combinatorial COMB4	35
5	Experiments	37
5.1	Experiments with Learning Algorithms	37
5.1.1	Parameter tuning	37
5.1.1.1	EXP3 exploration versus exploitation trade-off	38
5.1.1.2	Discounted UCB parameters	39
5.1.1.3	Sliding-window UCB parameters	41
5.1.2	Algorithm performance comparison	44
5.2	Experiments with Game-theoretic Strategies	47
5.2.1	Game-theoretic strategy with error	47
5.2.2	Motivating game-theoretic strategy	48
5.3	Performance of Combined Algorithms	49
5.3.1	Parameter tuning	49
5.3.2	COMB algorithms performance	51
5.4	Performance of Combined Algorithms with Motivating SSE	55
5.5	Combinatorial Combined Algorithms	56
5.5.1	Varying number of defenders	56
5.5.2	Scaling-up experiments	58
6	Conclusion	61
6.1	Future Work	62
	Appendix	64

A Content of DVD	65
A.1 Game Model	65
A.2 Algorithm for Solving Stackelberg Games	66

List of Figures

4.1	Chosen zones for the players in UCB synchronization	29
5.1	Defender online learning strategies against random fixed attacker types	45
5.2	Defender online learning strategies against adversarial attacker	46
5.3	Defender online learning strategies against Nash attacker	47
5.4	SSE strategies with different levels of error against adversarial attacker	48
5.5	COMB1 algorithm against adversarial attacker - τ tuning	50
5.6	COMB algorithms with 0.1 error against adversarial attacker	52
5.7	COMB algorithms with no error against adversarial attacker	53
5.8	Convergence of COMB algorithms with 0.1 error against adversarial attacker	54
5.9	COMB algorithms against adversarial attacker using motivating SSE	55
5.10	COMB algorithms against adversarial attacker, 0.1 error in SSE, varying number of defenders	57
5.11	COMB algorithms against adversarial attacker, 0.1 error in SSE, scaling up .	59

List of Tables

2.1	Payoff matrix of our analyzed game model	10
4.1	Payoff matrix of UCB synchronization game model	28
4.2	Game example of UCB synchronization	28
5.1	Tuning parameter γ of EXP3 algorithm against random attacker	38
5.2	Tuning parameter γ of EXP3 algorithm against adversarial attacker	38
5.3	Tuning parameter B of D-UCB algorithm against random attacker	39
5.4	Tuning parameter B of D-UCB algorithm against adversarial attacker	40
5.5	Tuning parameter γ of D-UCB algorithm against random attacker	40
5.6	Tuning parameter γ of D-UCB algorithm against adversarial attacker	41
5.7	Tuning parameter B of SW-UCB algorithm against random attacker	42
5.8	Tuning parameter B of SW-UCB algorithm against adversarial attacker	42
5.9	Tuning parameter τ of SW-UCB algorithm against random attacker	43
5.10	Tuning parameter τ of SW-UCB algorithm against adversarial attacker	43
5.11	Motivating SSE with decreased highest value	48
5.12	Motivating SSE with error with decreased highest value	49
5.13	Algorithm COMB1 τ parameter tuning against adversarial attacker	50
5.14	Algorithm COMB1 τ parameter tuning against adversarial attacker with changes	50

Chapter 1

Introduction

In the modern world we are more and more dependent on technology, which brings new vulnerabilities and increases the chance to be attacked or compromised by hackers, terrorists and other malicious groups. There is a huge demand to come up with efficient methods how to prevent these attacks. Recently it has been shown that game theory might be a very powerful tool to address such problems. Game theory is a well-known approach to solving various problems by analyzing interactions among intelligent agents. It has become recently very popular across many research fields such as economics, computer science, biology, electrical engineering, law and some others. Some new challenges appeared with increasing number of terrorist activities or drug trafficking. In all such problems there are limited resources, which need to be used effectively [29]. Several algorithms were deployed in real systems such as *Assistant for Randomized Monitoring Over Routes* (ARMOR), which was successfully deployed at the Los Angeles International Airport (LAX) in 2007 or *Intelligent Randomization in Scheduling* (IRIS) used to deploy air marshals on U.S. air carriers and others [29]. The game theory is often used in domains where there are limited resources and they need to be allocated efficiently. This is the case in many real world problems, where there are never enough resources to cover all possible threats. Another interesting application is discussed in [31], where the authors present new framework called *Protection Assistant for Wildlife Security* (PAWS), which improves wildlife ranger patrols.

In this thesis we review the recent literature on security games, we discuss published game models and algorithms and describe their limitations in real-world applications. We focus on a game model known as Stackelberg Security Game (SSG), which is used in many security domains. We study several approaches to solve SSG and analyze their limitations and usability.

Firstly we investigate online learning algorithms used for handling multi-armed bandit problem (MAB). MAB is a well-known concept in security games describing uncertainties in opponents' strategies. We focus on several known online learning algorithms such as UCB (Upper Confidence Bound), EXP3 (Exponential-weight algorithm for Exploration and Exploitation), non-stationary versions of UCB (Sliding-window UCB, Discounted UCB). We also study a combinatorial online learning algorithm COMB-EXP-1, which is a combinatorial version of EXP3 algorithm. Online learning algorithms have the ability to adapt to varying attackers or more precisely to varying attacker strategies. These algorithms learn the opponent strategy and respond as good as possible. However their limitation is quite poor performance at the beginning of the game, where they have no prior information about the at-

tacker strategy. Therefore it makes sense to somehow initialize the online learning algorithms and thus improve their performance.

Secondly we describe game-theoretic approach to solve SSG, where we focus on Nash and Stackelberg equilibrium strategy. We study usability of such approach and point out its limitations in the security games. We also investigate the possibilities of computing precise equilibria strategies in real-world security games and the impact of possible errors in the equilibria computation on the overall performance. A limitation of the game-theoretic approach is its inability to adapt to varying opponents. When using game-theoretic strategy we are limited to fixed strategy vector and cannot react to any changes in the opponent strategy.

Based on these two approaches to handle SSG, we derive combined algorithms. We propose 4 combined algorithms, which use EXP3 online learning algorithm with game-theoretic solution. The combined algorithms differ by a method of combining the game-theoretic solution with online learning algorithm and by adaptability to varying opponent and level of use of the game-theoretic solution. The combined algorithms address some of the previously described limitations of basic approaches, therefore they are able to adapt to non-stationary environments and have improved early performance. We also present combinatorial versions of combined algorithms. We empirically test these statements on many experiments with varying game settings.

We show these methods on a game model of Border Patrol, where we need to allocate the resources as good as possible to apprehend illegal trespassers. This model is motivated by [1], where the authors describe real-world security domain of border patrol.

1.1 Related Work

To the best of our knowledge, dealing with uncertainty in players' perception in Stackelberg security games (Section 2.3) is a very new research area and there are quite few scientific papers studying it. Some early papers study unrealistic or very simplistic assumptions e.g. player's perfect knowledge about opponent strategy. Nevertheless they are necessary first steps to develop more realistic models of Stackelberg games. We mainly base this thesis on [3], [2] and [7]. These papers differ by strength of simplifications of input assumptions. One of the first profound work on imprecise information in Stackelberg security games (SSG) is [3], where the authors state that most of the previous work on such topic assumed perfect knowledge of the defender and rationality of the attacker, which is the assumption for perfect knowledge SSG [17].

We firstly analyze a necessary background for this thesis and differences between approaches of mentioned papers. In [3] the imprecise information is modeled by limited number of attacker's observation of the defender distribution. The defender chooses his strategy first, the attacker then imprecisely observes it and decides which pure strategy he will play. The authors make a limiting assumption that the exact number of observations will be known to both players, this is justified by expert advice or intelligence. The authors also propose a heuristic approach to estimate the number of observations made by the attacker. Another assumption is that the defender knows the prior belief of the attacker about the defender's strategy. The authors propose an exact non-convex algorithm, so no solver can guarantee that the solution is optimal due to the non-convexity, therefore the authors also propose a convex approximation algorithm. The proposed algorithms are compared to Strong Stackelberg Equilibrium (SSE) (Section 2.5.3) and is shown their convergence to SSE with increasing

number of the observations; in our thesis we also compare all the algorithms to SSE. SSE is an important concept to compare with new algorithms in SSG model.

Succeeding work [2] addresses these not so realistic assumptions and proposes a model where the defender does not know the number of observations and the attacker does not know the defender strategy. They propose a stopping model, which tells the attacker when to stop observing the defender’s strategy and attack. The model uses Markov Decision Process (MDP). It is quite problematic to find the observation cost for the attacker, which is a critical parameter in the model. Based on this attacker’s strategy they propose an algorithm to get defender’s optimal strategy. Furthermore Yin et al. [32] propose a model with observational uncertainty; they assume a bounded difference between the defender’s strategy and the attacker’s estimate of it.

Another interesting approach from [7] directly reacts to (An et al. 2012, 2013) [3] and [2]. The authors state that the limited surveillance does not have to be an issue. Their model considers a zero-sum game. They found lower and upper bound for the difference between games, where the attacker has unlimited number of observations (knows exact defender’s distribution) and the case where the attacker has only τ observations. These bounds are shown to be quite tight. This does not hold for general-sum games. Mentioned paper [7] shows new perspective to limitations of proposed models with uncertainties in real applications. That is one of the reasons why we focus on slightly different approach to the studied problem in this thesis and focus only on one type of uncertainty in general-sum game.

In [19] the authors propose an algorithm, which finds a Stackelberg equilibrium strategy in a game with uncertain observability. They focus only on two cases, where first case is a full observability of leader’s strategy by the follower with some probability p_{obs} and the second case is that the follower is not able to observe the leader’s strategy with a probability $1 - p_{obs}$. Disadvantage of this model is a difficult determination of the number p_{obs} .

In [25] the authors focus on a model, which combines all possible uncertainties. They present 3-dimensional space of uncertainty in adversary payoff, uncertainty in adversary rationality and uncertainty in defender’s strategy. In our model we focus on uncertainty of the first type and we do not expect the defender to choose a different strategy than he is planning to due to some circumstances. We focus on uncertainty in defender’s estimate of attacker’s payoffs and we assume that the attacker is rational and chooses the action with the highest payoff. Although the attacker is rational, he has only an estimate of the defender’s strategy based on the observations. Proposed model by [25] can handle any type of uncertainty in the uncertainty space, but has limitation in scaling up to larger problems; this is caused by potentially large numbers of integer variables. Our model does not have this issue due to focusing only on one type of uncertainty, which is still well justifiable in practice. They also propose a scalable algorithm where they assume a rational attacker. The main difference between this paper and our thesis is in the ability of the proposed algorithms to be optimal. Their strategy is robust against various types of uncertainties but is limited to remain sub-optimal. However our proposed algorithms learn the opponent strategy and are able to reach the optimal strategy at the expenses of restricted robustness.

In [22] they analyze learning optimal Stackelberg strategies in Stackelberg games. The authors consider two game settings. Firstly the follower’s payoffs are not known and secondly the follower’s payoffs are known but the distribution over types is not. In our model we consider the first case. The latter case does not fit our model because we assume an adversary attacker who is very likely to choose his best response. They also mention its drawbacks that this framework might not be suitable for practical real-world applications due to the high cost

of learning phase.

Most of these models proposed by the mentioned literature focus on game-theoretic strategies and are not able to adapt to intelligent attacker, who can learn the defender strategy from the observations. In our previously published paper [18] we used online learning in security game model; we used several learning algorithms such as EXP3 [6], UCB [4], Sliding-window UCB [16]. In this thesis we want to address new approach to SSG where we make use of game-theoretic solution and online learning solution so we propose a set of combined algorithms. This approach enables the defender to learn the attacker's strategy and thus to adapt against intelligent attackers. We use several attacker types to test our proposed algorithm. Our model is designed as a fictitious play where each player observes the actions of the opponent and computes an estimate of the mixed strategy of the opponent. However the defender can observe only those actions of the attacker where the defender chooses the same action as the attacker, this feature comes from large domain of security games, where the defender can only observe unsuccessful attacks (successful apprehensions) e.g. border patrol problem [1]. This assumption is made without loss of generality because it is the worst-case for the defender. The attacker can make surveillance and observe all the actions of the defender. At each stage (round) a player updates his estimate of the opponents' strategy. In [24] they propose a security game model with incomplete information using standard fictitious play and stochastic fictitious play. They present a strategy for the players to reach Nash equilibria, if the error rates are known or a distance from Nash equilibria, if the error rates are unknown. Error rate means a level of the error in perception of other player strategy. We assume a general-sum game, where the attacker payoffs are based on the attacker action preferences.

The papers mentioned above mostly assume restricted conditions of the game, which is caused by a high complexity of the studied domain. In real-world applications there is a vast number of possible features, which can make the model too complex to deal with. Therefore the proposed models have some limitations, which we need to be aware of and deal with them appropriately. Generally, our goal is to propose a model as accurate as possible with feasible analysis. Our model represents another interesting approach, which is inspired by a real security domain of border patrol. Its advantage is that it requires minimum input information compared to mentioned papers. The proposed defender strategies by this thesis are able to reach the optimal strategy even though they start with arbitrarily imprecise information. Some of the papers mentioned above are necessary for formal analysis and for description of security domain pitfalls. In this thesis we propose quite a realistic model, which is easily described and makes some simplifications, which do not harm much the generality of the model. The main novelty of this thesis is the ability of the proposed methods to adapt to the attacker and learn from the observations while using game-theoretic solution. Some of the papers mentioned above show a theoretical model, how to learn enough from the distribution to commit to the optimal strategy. Closely related paper to our work is [31] where the authors propose a model, which can adapt to adversarial attacker and learn based on his previous behavior. However their model assumes a perfect knowledge about the defender strategy, which is not the case in this thesis. In our model the attacker has only an estimate of the defender strategy based on his surveillance of patrols.

1.2 Thesis Outline

Chapter 1 introduced the security domain and reviewed the current work on the topic. In Chapter 2 we introduce game theory concepts and we explain basic terms. We describe the Stackelberg Security Games and propose a game model, which we focus on in this thesis. We also mention game-theoretic approaches to solve a game. Chapter 3 describes learning in games and focus on the learning algorithms and multi-armed bandit problem. From Chapter 4 follows our own work and contribution where there is a description of our game model and used algorithms. Then we introduce the new combined algorithms. Chapter 5 focuses on various experiments to empirically show our results. Chapter 6 contains a conclusion of the thesis and possible future work.

Chapter 2

Game Theory Introduction

Game theory is a mathematical model of independent, self-interested agents who interact with each other. The original application was in economics but later on it has been used in several other domains like biology, psychology, political science or computer science. Game theory subdomain focuses on non-cooperative agents. In our work we study subdomain of non-cooperative game theory where the agents' interests conflict. The case where the agents' interests are same or similar is called coalitional or cooperative game theory.

2.1 Game Model

We describe mathematically the game model and related concepts.

2.1.1 Normal-form game

Normal-form game is the basic game representation also called the strategic form. Most of the other game representation can be transformed to the normal-form game. Normal-form game definition is taken from Shoham (2009) [28] page 56.

Definition 1. (Normal-form game) *A (finite, n-person) normal-form game is a tuple (N, A, x) , where:*

- *N is a finite set of n players, indexed by i :*
- *$A = A_1 \times \dots \times A_n$, where A_i is a finite set of actions available to player i . Each vector $a = (a_1, \dots, a_n) \in A$ is called an action profile;*
- *$x = (x_1, \dots, x_n)$ where $x_i : A \mapsto \mathbb{R}$ is a real-valued utility (also payoff or reward) function for player i .*

In the definition above the authors use originally¹ u as *utility* but we replace it with x because in the security games the utilities are often represented by letter x instead of u , which is the case in this thesis since we deal with the security games. Instances of the games are represented by matrices; in case of two players we have two-dimensional matrix. This matrix is called the *payoff matrix* where each row represents a possible action for player A and each column represents a possible action for player B. Example of a payoff matrix is the Table 2.1.

¹Taken definitions are in italic even in the cases where we slightly change the notation.

2.2 Game Concepts

We now describe the relation of payoffs between the players. There are two basic payoff structures, first one is called constant-sum game and the second is called general-sum game. We take the definition of the constant-sum game from [28] page 58.

Definition 2. (Constant-sum game) *A two-player normal-form game is constant-sum if there exist a constant c such that for each strategy profile $a \in A_1 \times A_2$ it is the case that $x_1(a) + x_2(a) = c$.*

A special case of the constant-sum game is a *zero-sum* game where $c = 0$. Zero-sum games are very widely used concepts in game theory, we can understand it as a situation where gain of the first player is equal to loss of the second player and vice versa.

We defined possible actions of the players and now we define how the players choose from them. There are two basic types of strategies, firstly *pure strategy* and secondly *mixed strategy*. Pure strategy is defined as a single action, which a player decides to play. Pure strategy profile is a choice of pure strategy for each agent in the game. More sophisticated strategy is that the player chooses to play randomly from the set of possible actions according to some probability distribution vector. We call this strategy the mixed strategy. Mixed strategy is often used strategy for the players in real-world games. The definition of the mixed strategy is taken from [28] page 59-60.

Definition 3. (Mixed strategy) *Let (N, A, x) be a normal-form game, and for any set X let $\Pi(X)$ be the set of all probability distributions over X . Then the set of mixed strategies for player i is $S_i = \Pi(A_i)$.*

And also the mixed-strategy profile

Definition 4. (Mixed-strategy profile) *The set of mixed-strategy profiles is the Cartesian product of the individual mixed-strategy sets, $S_1 \times \dots \times S_n$.*

A useful concept is the *support*, which is a subset of actions with non-zero probabilities. We denote $s_i(a)$ the probability that an action a_i is played in the mixed strategy s_i . Support definition is taken from [28] page 60.

Definition 5. (Support) *The support of a mixed strategy s_i for a player i is the set of pure strategies $\{a_i \mid s_i(a_i) > 0\}$.*

Since we defined the mixed strategy we can define the *expected payoff*. If we have a probability of playing each action in the action space for each player and we have the payoff matrix, which specify the payoff for each action and for each player we can compute the expected payoff. The definition is taken from [28] page 60.

Definition 6. (Expected payoff of a mixed strategy) *Given a normal-form game (N, A, x) the expected payoff \bar{x}_i for player i of the mixed-strategy profile $s = (s_1, \dots, s_n)$ is defined as*

$$\bar{x}_i(s) = \sum_{a \in A} x_i(a) \prod_{j=1}^n s_j(a_j).$$

2.3 Stackelberg Security Games

We define Stackelberg security game according to [17]. The game has two players, the defender Θ and the attacker Ψ . In security games we usually do not have individuals playing against each other but rather groups of people who have similar or same goal. These groups can represent terrorists, hackers, etc. on the attacker side and officers, authorities, security units etc. on the defender side. These groups use a joint strategy so we can think of the group as an individual player with several resources. The defender has a set of pure strategies, denoted $\sigma_\Theta \in \Sigma_\Theta$ and the attacker has a set of pure strategies, denoted $\sigma_\Psi \in \Sigma_\Psi$. We consider a mixed strategy, which allows to play a probability distribution over all pure strategies, denoted $\delta_\Theta \in \Delta_\Theta$ for the defender and $\delta_\Psi \in \Delta_\Psi$ for the attacker. We define payoffs for the players over all possible joint pure strategy outcomes by $\Omega_\Theta : \Sigma_\Psi \times \Sigma_\Theta \rightarrow \mathbb{R}$ for the defender and $\Omega_\Psi : \Sigma_\Theta \times \Sigma_\Psi \rightarrow \mathbb{R}$ for the attacker. The payoffs for the mixed strategies are computed based on the expectations over pure strategy outcomes.

Important concept in Stackelberg security games is the idea of a leader and a follower. This concept is the main difference from the normal-form game. The defender is considered as the leader and the attacker as the follower. The leader plays first then plays the attacker who can fully observe the defender strategy before acting. This is quite a strong assumption, as we will see in our further analysis. This represents very adversarial and intelligent attacker who can fully observe the defender's strategy before deciding how to act. In case of naive defender or not sophisticated enough, the attacker can exploit the defender's strategy. This is a vulnerability, which is modeled by Stackelberg security game model. Formally we can describe the attacker's strategy as a function which chooses a mixed distribution over pure strategies for any defender's strategy: $F_\Psi : \Delta_\Theta \rightarrow \Delta_\Psi$.

2.4 Analyzed Game Model

In this thesis we propose a security game model for border patrol resource allocation. The model is inspired by real application [1]. This model comes from the concept of Stackelberg security games [17]. We assume the leader to be the defender whose goal is to apprehend any illegal immigrant crossing the border. In the real application the defender can be Office of Border Patrol (OBP). The follower is an attacker who is trying to cross the border without being apprehended. We assume a repeated game with imperfect information. The defender is playing a version of multi-armed bandit problem (Section 3.1) and the attacker is playing a version of a fictitious play (Section 3.5). The defender does not know the attacker strategy, he can only observe received payoffs. So he receives only payoffs from the zones he visited. The attacker can observe the whole patrol history, so he knows how many times the defender visited particular zone. In our model the attacker has a zone preference vector, which describes how easy or how hard is to cross a particular zone. This can be caused by a difficult access to the zone. The defender has no preferences for the zones. We analyze this game model and propose suitable algorithms to approach it. In our basic model we divide the whole border into 8 zones. These zones can be seen as arms of multi-armed bandit problem (described in Section 3.1) or as actions in standard game model definition. In the model we do experiments with random attacker zone preference vectors, thus each experiment use different payoff matrix. We provide here an example of such game to better understand the model. The defender payoffs are constant across all the experiments and all the zones. However we now present an

example of payoff matrix to better understand our general model.

Let's assume a zone preference vector for the attacker

$$v^\Psi = \{0.1, 0.2, 0.05, 0.3, 0.1, 0.05, 0.05, 0.15\}$$

We define rewards for the players. We distinguish if the player is covered by another player or not (covered means they choose the same zone in one round). If the defender is covered it means he apprehends the attacker, so he gets reward 1. If the defender is in different zone than the attacker then the defender does not apprehend the attacker and gets reward 0. The attacker uses a penalty of being caught P^Ψ which is 0.5 and computes his reward for a zone i according to

$$x_i^\Psi = (v_i^\Psi - P^\Psi * c_i^\Psi) \quad (2.1)$$

where c_i^Ψ is probability of being caught for the attacker which is 1 in case of being covered and 0 in case of being not covered. In the experiments the probability of being caught is computed from patrol history and assigns a value to each zone, how probable is that the defender visits this particular zone. So for this example game we can write the rewards

covered defender payoff:

$$x_c^\Theta = \{1, 1, 1, 1, 1, 1, 1, 1\}$$

uncovered defender payoff:

$$x_u^\Theta = \{0, 0, 0, 0, 0, 0, 0, 0\}$$

covered attacker payoff:

$$x_c^\Psi = \{-0.4, -0.3, -0.45, -0.2, -0.4, -0.45, -0.45, -0.35\}$$

uncovered attacker payoff:

$$x_u^\Psi = \{0.1, 0.2, 0.05, 0.3, 0.1, 0.05, 0.05, 0.15\}$$

and payoff matrix in normal-form notation

zone	1	2	3	4	5	6	7	8
1	1, -0.4	0, 0.2	0, 0.05	0, 0.3	0, 0.1	0, 0.05	0, 0.05	0, 0.15
2	0, 0.1	1, -0.3	0, 0.05	0, 0.3	0, 0.1	0, 0.05	0, 0.05	0, 0.15
3	0, 0.1	0, 0.2	1, -0.45	0, 0.3	0, 0.1	0, 0.05	0, 0.05	0, 0.15
4	0, 0.1	0, 0.2	0, 0.05	1, -0.2	0, 0.1	0, 0.05	0, 0.05	0, 0.15
5	0, 0.1	0, 0.2	0, 0.05	0, 0.3	1, -0.4	0, 0.05	0, 0.05	0, 0.15
6	0, 0.1	0, 0.2	0, 0.05	0, 0.3	0, 0.1	1, -0.45	0, 0.05	0, 0.15
7	0, 0.1	0, 0.2	0, 0.05	0, 0.3	0, 0.1	0, 0.05	1, -0.45	0, 0.15
8	0, 0.1	0, 0.2	0, 0.05	0, 0.3	0, 0.1	0, 0.05	0, 0.05	1, -0.35

Table 2.1: Payoff matrix of our analyzed game model

In the payoff matrix in Table 2.1 the defender is the row player and the attacker is the column player.

2.5 Equilibrium in Game

2.5.1 Dominant strategy

Intuitively if one strategy dominates another it means that a player gets a higher payoff if he plays the first strategy instead of the another. There are several types of dominance. We take a dominant strategy definition from [28] page 78.

Definition 7. (Domination) Let s_i and s'_i be two strategies of player i , and S_{-i} the set of all strategy profiles of the remaining players. Then

1. s_i strictly dominates s'_i if for all $s_{-i} \in S_{-i}$, it is the case that $x_i(s_i, s_{-i}) > x_i(s'_i, s_{-i})$.
2. s_i weakly dominates s'_i if for all $s_{-i} \in S_{-i}$, it is the case that $x_i(s_i, s_{-i}) \geq x_i(s'_i, s_{-i})$, and for at least one $s_{-i} \in S_{-i}$, it is the case that $x_i(s_i, s_{-i}) > x_i(s'_i, s_{-i})$.
3. s_i very weakly dominates s'_i if for all $s_{-i} \in S_{-i}$, it is the case that $x_i(s_i, s_{-i}) \geq x_i(s'_i, s_{-i})$.

Definition 8. (Dominant strategy) A strategy is strictly (resp. weakly, very weakly) dominant for an agent if it strictly (weakly, very weakly) dominates any other strategy for the agent.

2.5.2 Nash equilibrium

We define Nash equilibrium, which is a necessary basic concept in game theory. This concept was introduced by John Nash in 1951 [23]. We first need to introduce player's best response and then we can introduce the Nash equilibrium, the definitions are taken from Shoham [28] page 62.

Definition 9. (Best response) Player i 's best response to the strategy profile s_{-i} is a mixed strategy $s_i^* \in S_i$ such that $x_i(s_i^*, s_{-i}) \geq x_i(s_i, s_{-i})$ for all strategies $s_i \in S_i$.

Best response is player's strategy concept, which does not have to be unique. There is usually more than one best response in a game, otherwise the best response is also pure strategy. The player must be indifferent among best responses if there are more than one.

Definition 10. (Nash equilibrium) Nash equilibrium is a strategy profile $s = (s_1, \dots, s_n)$ if for all agents i , s_i is a best response to s_{-i} .

John Nash introduced the Nash equilibrium in 1951 and also proposed this theorem taken from [23].

Theorem 1. Every game with a finite number of players and action profiles has at least one Nash equilibrium.

This is a very important conclusion for game theory. In the security games there arose a desire to find such an equilibrium for which the players act optimally. We stated that the best response does not have to be unique which means that there are possibly multiple Nash equilibria in a game. This introduces a problem how to choose among multiple Nash equilibria. However there is no such problem in Stackelberg equilibria described in the next section.

Example of NE in our game

We compute the Nash equilibrium from the payoff matrix in Table 2.1. For the defender we get vector

$$s^\Theta = \{0.06, 0.26, 0, 0.46, 0.06, 0, 0, 0.16\}$$

and Nash equilibrium mixed strategy for the attacker is

$$s^\Psi = \{0.2, 0.2, 0, 0.2, 0.2, 0, 0, 0.2\}$$

We call those zones, which are played with non-zero probability by the defender *active*. We can clearly see that if the defender and the attacker play according to these strategies in our game model described in 2.4, the expected defender payoff for such game is $0.2 * T$ where T is the number of rounds played. On this example we can also demonstrate that it is not favorable for any player to change his strategy.

2.5.3 Stackelberg equilibrium

Stackelberg equilibrium is a refinement of Nash equilibrium, which is a strategy profile where no player can gain by unilaterally deviating to another strategy. We define concept of Stackelberg equilibrium (SE) according to [17] and we use the terminology described in section 2.3. This idea was firstly introduced in economy field where it described the reaction between agents. SE is a version of subgame perfect equilibrium where each player chooses a best response in any subgame of the original game. SE eliminates some Nash equilibrium strategies, which are not optimal in Stackelberg game. Subgame perfection is not a guarantee that there exists a unique SE. This is caused by possible indifference of the follower among a set of targets. To address this issue there are two concepts of SE called *strong Stackelberg equilibrium* and *weak Stackelberg equilibrium*. The strong SE assumes that in case of indifference between targets the follower chooses the optimal strategy for the leader and the weak SE assumes that the follower choose the worst strategy for the leader. Concept of strong SE avoids the problem of equilibrium selection, which is a very discussed problem for Nash equilibrium. A strong SE exists in every Stackelberg game but a weak SE might not. The leader can motivate the desired strong equilibrium by choosing a strategy, which is arbitrary close to the equilibrium. This makes the follower strictly choosing the preferred strategy. We take the definition from [17] where the authors define strong SE

Definition 11. (Strong Stackelberg equilibrium) A pair of strategies (δ_Θ, F_Ψ) form a Strong Stackelberg Equilibrium (SSE) if they satisfy the following:

1. The leader plays a best response:

$$\Omega_\Theta(\delta_\Theta, F_\Psi(\delta_\Theta)) \leq \Omega_\Theta(\delta'_\Theta, F_\Psi(\delta'_\Theta)) \forall \delta'_\Theta \in \Delta_\Psi$$
2. The follower plays a best response:

$$\Omega_\Psi(\delta_\Theta, F_\Psi(\delta_\Theta)) \leq \Omega_\Psi(\delta_\Theta, \delta_\Psi) \forall \delta_\Theta \in \Delta_\Theta, \delta_\Psi \in \Delta_\Psi$$
3. The follower breaks ties optimally for the leader:

$$\Omega_\Theta(\delta_\Theta, F_\Psi(\delta_\Theta)) \leq \Omega_\Theta(\delta_\Theta, \delta_\Psi) \forall \delta_\Theta \in \Delta_\Theta, \delta_\Psi \in \Delta_\Psi^*(\delta_\Theta),$$
 where $\Delta_\Psi^*(\delta_\Theta)$ is the set of follower best responses as defined above.

Example of SSE in our game

The SSE of our game model described in 2.4 is the same as the NE. So for the defender the mixed strategy is

$$s^\Theta = \{0.06, 0.26, 0, 0.46, 0.06, 0, 0, 0.16\}$$

and SSE mixed strategy for the attacker is

$$s^\Psi = \{0.2, 0.2, 0, 0.2, 0.2, 0, 0, 0.2\}$$

In the attacker strategy we can see that there are ties in the probability distribution. According to the definition of Stackelberg security game and SSE the attacker breaks ties in favor to the defender so he always plays zone number 4. From this deduction it is straightforward that the expected defender payoff is $0.46 * T$, where T is the number of rounds played.

Chapter 3

Learning in Games

In this chapter we focus on learning in games. In [33] they describe properties of online learning in games. Learning in two-player games is quite a complex problem because the learning itself changes the thing to be learned. Let's consider player 1 and player 2; player 1 wants to learn player's 2 strategy and he is learning by what he learned so far and what he is going to learn next. Also the player 2 can observe what player 1 learned so far. Thus, the strategy of player 2 can change as a result of player's 1 attempt to learn it. Of course the same fact holds for the player 2. This loop property is inevitable and makes the model quite complex to analyze and its behavior rather unintuitive. In general this property can cause that one of the players or both players cannot learn the optimal strategy at all. There is an example of such case in Section 4.2.1, where we describe learning algorithm synchronization, which disable the opponent to learn the optimal strategy. This analysis shows that online learning has its restrictions we should be aware of.

Nevertheless many game models rely on equilibrium analysis and use Nash equilibrium or a refinement of it. In many real world applications there is not possible to obtain equilibrium strategy at the beginning due to uncertainties of payoff matrices or opponent's strategies. However equilibrium can be obtained by learning and adaptation. In some games players choose a strategy, which maximize their payoff given beliefs that they obtained from past rounds of the game. We describe a fictitious play for example, which is discussed in Section 3.5. In [15] they discuss a fictitious play and other types of learning in games. Another type of learning system deals with multi-armed bandit problem.

3.1 Multi-armed Bandit Problem

Multi-armed bandit problem (MAB) is one of the most basic concepts in online learning. This problem was extensively studied in statistics and appears constantly across various fields such as artificial intelligence and others. We describe multi-armed bandit problem according to [8] but it was originally proposed by Robbins [26].

MAB generally deals with a tradeoff between exploration and exploitation, which is a well-known decision problem where we choose either the best explored option so far (exploitation) or try a new unexplored option (exploration), which might or might not give us better result. One can show this problem on everyday life's choices; should one go to a restaurant he well knows and is sure he will get a good service and a decent meal or should he try some new restaurant with uncertain outcome, therefore he can be lucky and get even better service and

meal than in his favorite restaurant or he can have a bad luck and get a really bad meal and service. If he would only stick to restaurants he knows, he is not ever able to discover any new great restaurant. Thus, he needs to come up with a good strategy - exploration versus exploitation tradeoff to get an optimal outcome - a great service and a decent meal. MAB problem appears in various domains like ad placement, website optimization, security games, etc. The name *bandit* refers to a slang term in American English for a slot machine - a *one-armed bandit*. The analogy to the problem is a player in a casino playing multiple slot machines with different payoff distribution. He must repeatedly decide where he puts his next coin. MAB is a sequential allocation problem with a set of actions (bandits). We can see it as a sequential decision making with limited information. Even though it is quite weak setting for learning, it is very realistic and used in many real world applications [13].

Based on the nature of reward process there are three basic MAB formalizations: stochastic, adversarial and Markovian. For these three models there were presented effective algorithms: UCB algorithm for the stochastic case, EXP3 algorithm for the adversarial case and an algorithm using Gittins indices for the Markovian case. We focus on the first two cases. We take the following definitions and equations from [8].

3.1.1 Regret

We want to analyze a performance of strategies for MAB. To do that we compare a chosen strategy with the optimal strategy. We need to quantify a difference between playing an arm chosen by the strategy and playing the optimal arm, therefore we define a *regret*.

$$R_n = \max_{i=1,\dots,K} \sum_{t=1}^n X_{i,t} - \sum_{t=1}^n X_{I_t,t} \quad (3.1)$$

where R_n is the regret after n plays, $K \geq 2$ is a number of arms, sequence $X_{i,1}, X_{i,2}$ are the unknown rewards of each arm $i = 1, \dots, K$. At each time step $t = 1, 2, \dots$ we chose an arm I_t and obtain the associated reward $X_{I_t,t}$.

We also assume a stochastic strategy so we define the *expected regret*

$$\mathbb{E}[R_n] = \mathbb{E} \left[\max_{i=1,\dots,K} \sum_{t=1}^n X_{i,t} - \sum_{t=1}^n X_{I_t,t} \right] \quad (3.2)$$

and the *pseudo-regret*

$$\bar{R}_n = \max_{i=1,\dots,K} \mathbb{E} \left[\sum_{t=1}^n X_{i,t} - \sum_{t=1}^n X_{I_t,t} \right] \quad (3.3)$$

In both equations the expected value is taken from a random draw from the strategy distribution. The pseudo-regret is a weaker notion of regret than the expected regret so we can write $\bar{R}_n \leq \mathbb{E}[R_n]$.

3.1.2 Stochastic bandit problem model

Each arm in the MAB model $i = 1, \dots, K$ corresponds to an unknown probability distribution p_i from $[0, 1]$. The stochastic bandit problem is defined

Known parameters: number of arms K and number of rounds n
Unknown parameters: arms prob. distributions p_1, \dots, p_K in $[0, 1]$
for $t = 1, \dots, n$ **do**
 (1) the player chooses arm $I_t \in 1, \dots, K$
 (2) having I_t the system draws the reward $X_{I_t}, t \sim p_{I_t}$ independently from the distribution and reveals it to the player.
end

Algorithm 1: The stochastic bandit problem model

We also define the mean reward μ_i for each arm distribution. So we define optimal mean reward

$$\mu^* = \max_{i=1, \dots, K} \mu_i$$

and

$$i^* \in \arg \max_{i=1, \dots, K} \mu_i$$

Now we can write the pseudo-regret as

$$\bar{R}_n = n\mu^* - \sum_{t=1}^n \mathbb{E}[\mu_{I_t}] \quad (3.4)$$

3.1.3 Adversarial bandit problem model

Now we assume the adversary scenario where the multi-armed bandit acts maliciously and wants the player to maximize his regret (minimize his reward), however the bandit cannot set the rewards to zero because he wouldn't attract any players to play it (here is the analogy to a casino). We call such bandit adversary or opponent. We differ two variants of adversaries, firstly an *oblivious* adversary whose distributions are independent to the player's actions, secondly and more commonly we have a *non-oblivious* adversary who can adapt to the player's past behavior. We focus on the latter variant of the adversary. The regret analysis for this model setting is based on the connection between regret minimization and game-theoretic equilibria in a game. In game-theoretic equilibrium the player has no need to change his strategy if the opponent observes it and reacts to it, which is the same property as in the regret minimization definition. So we define the adversarial bandit model as

Known parameters: number of arms K and number of rounds n
for $t=1, 2, \dots$ **do**
 (1) the player chooses an arm $I_t \in 1, \dots, K$, he can use some extra random exploration;
 (2) at the same instance the adversary (opponent) sets the distribution reward vector $d_t = (d_{1,t}, \dots, d_{K,t}) \in [0, 1]^K$, he can also use some extra randomization;
 (3) the player obtains the reward $d_{I_t,t}$, but he does not observe the other arms' rewards.
end

Algorithm 2: Adversarial bandit problem model

The model of adversarial multi-armed bandit developed in time. Firstly the model supposed a payoff matrix to be known to the player and also the player knew the opponent's

moves. Later versions of the model considered a repeated unknown game, where the player only observes his own payoffs. This is exactly the problem of non-oblivious adversarial bandit. In [4] they showed the connection to stochastic bandits by introducing new term of non-stochastic multi-armed bandit problem.

3.2 Combinatorial Bandits

In our experiments we scale up our model and analyze the case where there are more defender resources in each round than just one. This setting is called the combinatorial bandit case and we present a combinatorial learning algorithm. This is a special concept used in Stackelberg Security Games where we have a strategy vector from which we choose k actions to be played where k is the number of resources. There is a subset of single arms with unknown distributions which form a *super arm*. In each round a super arm is played, therefore each single arm in the super arm is played and the player receives a sum of rewards from all the single arms contained in the super arm. Combinatorial bandit case uses similar concepts as multi-armed bandit case as regret and others. The problem is well described in [9] where the authors propose a combinatorial learning algorithm based on EXP3 algorithm called *ComBand*. This algorithm is rather complex, therefore we use for our model algorithm *COMB-EXP-1* presented in [11] which is strongly inspired by ComBand algorithm. They also present a combinatorial version of UCB *CombUCB* algorithm suitable for stochastic combinatorial bandits. However we focus on adversarial combinatorial bandits for which COMB-EXP-1 is designed.

3.3 Convergence of Learning Algorithms

Generally we differ properties of learning algorithms between zero-sum games and general-sum games. Learning algorithms are shown to converge to Nash equilibrium in zero-sum games as shown in [13]. In this settings the learning algorithms are known to have a decreasing regret. For the general-sum games the learning algorithms converge to a more general equilibrium than the Nash equilibrium if the regret is minimized explicitly. In the paper they describe several learning algorithm types with respect to the game setting. They vary by the level of information each player gets about the opponent strategy. In our work we focus on a variant of multi-armed bandit problem for the defender where a player sample the distribution of opponent's strategy and update his belief according to the received payoffs. For the attacker we assume a variant of a fictitious play. Learning in multi-armed bandit setting or in a fictitious play setting does not necessarily converge to the Nash equilibrium in a general-sum game.

3.4 Learning Algorithms

We present two main learning algorithms for multi-armed bandit problem UCB and EXP3 algorithm. UCB plays a pure strategy because of its deterministic nature, however EXP3 has a stochastic strategy and is more suitable for mixed strategy equilibria.

3.4.1 Upper Confidence Bound (UCB) algorithm

This algorithm was presented in [4]. As we described in section about multi-armed bandit problem we face the exploration versus exploitation dilemma. Concept of a regret is highly used measure for analyzing learning algorithms. Lai and Robins [20] showed that a regret for multi-armed bandit problem has to grow at least logarithmically in the number of plays. Since they presented this proof many learning algorithms with logarithmic regret were derived. The Upper Confidence Bound (UCB) achieves logarithmic regret uniformly over number of rounds and without any preliminary knowledge about the reward distributions. They just assume that the support is in range $[0, 1]$.

Deterministic policy: UCB

Initialization: Play each action once.

for $t = 1, \dots, n$ **do**

| $a_t = \arg \max_j \bar{x}_j + \sqrt{\frac{2 \ln n}{n_j}}$

end

Algorithm 3: Pseudo-code of algorithm UCB

where t is the number of round, a_t is the action we play in the round t , \bar{x}_j is the mean reward obtained from action j , n_j is the number of times action j has been played so far and n is the overall number of rounds played so far.

In the model the parameter K is the number of machines or actions depending on domain.

Theorem 1. (Expected regret of UCB) For all $K > 1$, if policy UCB is run on K machines having arbitrary reward distributions P_1, \dots, P_K with support in $[0, 1]$, then its expected regret after any number n of plays is at most

$$\mathbb{E}[R] \leq \left[8 \sum_{i: \mu_i < \mu^*} \left(\frac{\ln n}{\Delta_i} \right) \right] + \left(1 + \frac{\pi^2}{3} \right) \left(\sum_{j=1}^K \Delta_j \right)$$

where μ_1, \dots, μ_K are the expected values (rewards) of P_1, \dots, P_K . And

$$\Delta_i = \mu^* - \mu_i$$

where μ^* is the maximal element (reward) in the model.

3.4.2 EXP3 algorithm

One of the widely used learning algorithms is *Exponential-weight algorithm for Exploration and Exploitation* called EXP3. This algorithm is derived from algorithm **Hedge** introduced in [14]. The EXP3 algorithm was described and analyzed in [6] or before in [5].

The parameter γ is called the exploration rate and sets how much the algorithm will explore the action space. The standard setting is $\gamma = 0.1$ which means 10% of exploration. Vector \mathbf{w} represents the estimated payoff of each action. The player uses the probability distribution vector \mathbf{p} as his strategy. This algorithm tends to be numerically unstable which is caused by computing the weights in step 4 in Algorithm 4. So under some circumstances there can be a problem with overflowing the number format. Due to this reason we use in

Parameters: Real $\gamma \in (0, 1]$
Initialization: $w_i(1) = 1$ for $i = 1, \dots, K$.
for each $t = 1, 2, \dots$ **do**
 1. Set

$$p_i(t) = (1 - \gamma) \frac{w_i(t)}{\sum_{j=1}^K w_j(t)} + \frac{\gamma}{K} \quad i = 1, \dots, K$$

 2. Draw i_t randomly accordingly to the probabilities $p_1(t), \dots, p_K(t)$
 3. Receive reward $x_{i_t}(t) \in [0, 1]$.
 4. **for** $j = 1, \dots, K$ **do**

$$\hat{x}_j(t) = \begin{cases} x_j(t)/p_j(t) & \text{if } j = i_t; \\ 0 & \text{if otherwise,} \end{cases}$$

$$w_j(t+1) = w_j(t) \exp(\gamma \hat{x}_j(t)/K)$$

 end
end

Algorithm 4: Pseudo-code of algorithm EXP3

our implementation of the model a slightly modified algorithm of EXP3 which is numerically stable, presented in [12]. The probability distribution is defined

$$p_i = \frac{\gamma}{K} + \frac{1 - \gamma}{\sum_{j \in K} e^{(s(j) - s(i))\eta}} \quad (3.5)$$

where $s(i)$ is the sum of rewards from previously selecting arm i , each divided by the probability of selecting i on that trial. η and γ are constant parameters.

They analyze the algorithm EXP3 in [8]. The authors use a pseudo-regret to define the upper bound, we defined the pseudo-regret concept in section on multi-armed bandit problem. They present a pseudo-regret of EXP3 algorithm

Theorem 2. (Pseudo-regret of EXP3) If EXP3 exploration rate

$\gamma = \sqrt{\frac{2 \ln K}{nK}}$, then

$$\bar{R}_n \leq \sqrt{2nK \ln K} \quad (3.6)$$

This theorem holds for the situation where the player knows the number of rounds n . If he does not know the number of rounds, we get the anytime version of the algorithm

Theorem 3. (Pseudo-regret of EXP3 - anytime version) If EXP3 exploration rate

$\gamma_t = \sqrt{\frac{\ln K}{tK}}$, then

$$\bar{R}_n \leq 2\sqrt{nK \ln K} \quad (3.7)$$

Proof of these two theorems can be found in [8]. Relation between the expected regret and the pseudo-regret is $\bar{R}_n \leq \mathbb{E}[R]$, so the upper bound on the pseudo-regret does not imply a bound on the expected regret. In [5] they prove an upper bound for the expected regret for EXP3 with suitably chosen parameters of the used algorithm.

Theorem 4. (Expected regret of EXP3) If the exploration rate is ([5], Corollary 4.2)

$$\gamma = \min \left\{ 1, \sqrt{\frac{K \ln K}{(e-1)n}} \right\} \quad (3.8)$$

and $\eta = \frac{\gamma}{K}$. Then for such setting the upper bound on the expected regret is

$$\mathbb{E}[R] \leq 2\sqrt{e-1}\sqrt{nK \ln K} \leq 2.63\sqrt{nK \ln K} \quad (3.9)$$

In [5] they prove this theorem. In our implementation of the model we use parameter $\eta = \frac{\gamma}{K}$.

3.4.3 Non-stationary UCB versions

We present a version of standard UCB algorithm called *Discounted UCB (D-UCB)* and *Sliding-window UCB (SW-UCB)*. UCB algorithm is designed to work well in stationary environment which assume that the distribution of the rewards do not change over time. It is suitable for our model to have some tool for non-stationary environment where the distributions of the rewards can change during the game. D-UCB and SW-UCB have a better ability to adapt to such a changing environment. We describe the algorithms according to [16].

The non-stationary environment is defined by the rewards $\{X_t(i)\}_{t \leq 0}$ for arm i at time step t , these rewards are sequences of independent random variables from different distributions which may vary during the time. These distributions are unknown to the player. We denote $\mu_t(i)$ the expectation of the reward $X_t(i)$ for arm i . In [16] they assume *abruptly changing environment* which means that the distributions remain constant and at unknown time instant the distributions change abruptly. We call these time instants *breakpoints*. We define a number of these breakpoints by Υ_T before time step T .

3.4.3.1 Discounted UCB algorithm

The main difference between the standard UCB and the discounted UCB (D-UCB) is that the algorithm uses a discounted value of data from the previous rounds to calculate the estimated average rewards. The discounted UCB relies on a discount factor $\gamma \in (0, 1)$. At time step t we get discounted value of rewards from the previous rounds with giving more weight to more recent observations. D-UCB chooses a zone, which maximize the sum of exploitation and exploration part. The exploitation part of the D-UCB formula is a discounted empirical average

$$\bar{X}_t(\gamma, i) = \frac{1}{N_t(\gamma, i)} \sum_{s=1}^t \gamma^{t-s} X_s(i) \mathbb{1}_{\{I_s=1\}} \quad (3.10)$$

where

$$N_t(\gamma, i) = \sum_{s=1}^t \gamma^{t-s} \mathbb{1}_{\{I_s=1\}} \quad (3.11)$$

where $X_s(i)$ is a reward in time step s of i th zone and the indicator function returns a value of one if the chosen zone in the time step s is equal to i th zone, and zero otherwise.

The exploration part of the formula which is called the discounted padding function is defined

$$c_t(\gamma, i) = B \sqrt{\frac{\log n_t(\gamma)}{N_t(\gamma, i)}} \quad (3.12)$$

where

$$n_t(\gamma) = \sum_{i=1}^K N_t(\gamma, i) \quad (3.13)$$

B is a suitably chosen parameter according to the environment properties. Algorithm chooses an action i which maximize the argument

$$a(t) = \arg \max_{1 \leq i \leq K} \bar{X}_t(\gamma, i) + c_t(\gamma, i) \quad (3.14)$$

3.4.3.2 Sliding-window UCB algorithm

The main difference from the standard UCB is that the sliding-window UCB (SW-UCB) algorithm uses a fixed window of data from the previous rounds to calculate the estimated average rewards. At time step t we get average of rewards not from the whole history but only the τ previous rounds. SW-UCB chooses a zone, which maximize the sum of exploitation and exploration part. The exploitation part of the UCB formula is a local average reward

$$\bar{X}_t(\tau, i) = \frac{1}{N_t(\tau, i)} \sum_{s=t-\tau+1}^t X_s(i) \mathbb{1}\{I_s = i\} \quad (3.15)$$

where

$$N_t(\tau, i) = \sum_{s=t-\tau+1}^t \mathbb{1}\{I_s = i\} \quad (3.16)$$

is the number of times playing arm i in τ previous rounds.

The exploration part (padding function) is defined as

$$c_t(\tau, i) = B \sqrt{\frac{\log(t \wedge \tau)}{N_t(\tau, i)}} \quad (3.17)$$

where $(t \wedge \tau)$ denotes the minimum of two arguments and τ is a constant. B is a constant, which should be tuned appropriately to the environment. Algorithm chooses an action i which maximize the argument

$$a(t) = \arg \max_{1 \leq i \leq K} \bar{X}_t(\tau, i) + c_t(\tau, i) \quad (3.18)$$

In [16] the authors show a proof for upper bounds on regrets for these two algorithms and state that the algorithms' upper bounds on regret are $O(\sqrt{T \log(T)})$.

3.4.4 Combinatorial EXP3 algorithm

We use algorithm COMB-EXP-1 presented in [11]. This algorithm is a combinatorial version of EXP3 algorithm and is suitable for adversarial combinatorial bandit problem described in Section 3.2.

COMB-EXP-1 algorithm

Initialization: Start with the distribution $q_0 = \mu^0$ and set $\eta = \sqrt{\frac{2k \log z}{zT}}$

for all $n \geq 1$ **do**

1. Sample k actions from vector $p_{n-1} = kq_{n-1}$.
2. Obtain the reward vector $X_i(n)$ for all chosen actions i .
3. Compute the vector $\bar{X}_i(n) = \frac{1-X_i(n)}{kq_{n-1}(i)}$ for all chosen actions i and assign 0 to all other not chosen actions.
4. Update $\bar{q}_n(i) = q_{n-1}(i) \exp(-\eta \bar{X}_i(n))$.
5. Compute q_n to be a projection of \bar{q}_n onto the set \mathcal{P} using KL divergence.

end

Algorithm 5: Combinatorial EXP3 learning algorithm

where μ_0 is an uniform distribution vector over all actions (zones), k is the number of resources the defender has, z is the number of actions (zones) and T is the total number of rounds of the game. We describe a sampling method for step 1 in the next section. In Section 4.2.2 we propose a method for finding the projection for step 5 in Algorithm 5. Similarly to non-combinatorial EXP3 algorithm described in Section 3.4.2, the COMB-EXP-1 algorithm tends to be numerically unstable, which is caused by step 3, respectively step 4 in Algorithm 5. We prevent this instability in our implementation by adding an uniform vector with very small values (10^{-7}) to the strategy vector q , which bounds the possible expected reward \bar{X} .

The regret for COMB-EXP-1 is $O(m\sqrt{zT \log(z/k)})$ proven in [11].

3.4.4.1 Combinatorial sampling

We have a vector p from which we want to sample k actions. The vector p sums up to k and no its value is greater than 1. We use combinatorial sampling, which is described in [30] which the authors call *Comb Sampling*. From vector p we create a new cumulative sum vector. For each integer $j \in (1, J)$, let $X_j = \sum_{i < j} x_i$. Based on that we also define interval vector $I_j = [X_j, X_j + x_j]$, which form a disjoint cover of interval $[0, k]$, because $\sum_i x_i = k$. To sample from vector p we pick a number y from interval $[0, k)$ uniformly at random. Now we select indices of intervals which contain points $y, y+1, \dots, y+k-1$. This procedure samples exactly k actions.

3.5 Fictitious Play

Fictitious play is a well-known type of learning concept. It is based on beliefs about opponent's strategy. Having these beliefs the player makes rational decisions. The original model of fictitious play assumes opponent with stationary mixed strategy so that the player can have reliable beliefs about opponent's strategy based on the empirical frequencies of the opponent's play. Fictitious play (game) was proven to converge to Nash equilibrium in zero-sum game by Robinson (1951) [26]. For general-sum games Shapley showed [27] that fictitious play does

not generally converge. We describe the concept according to Levin [15]. We show a simple example of fictitious play.

We have two players $i = 1, 2$ who play the game G at times $t = 0, 1, 2, \dots$. We define $\eta_i^t(a)$ to be a number of times player i has observed action a in the past up to time t . We can also assume $\eta_i^0(a)$ to be a fictitious past, which we obtain at the beginning of the game. We consider an example of payoff matrix

	L	R
U	3,3	0,0
D	4,0	1,1

For the fictitious past we can have $\eta_1^0(U) = 3$ and $\eta_1^0(D) = 5$. So if the player 1 plays U, U, D in the first three rounds, then we get $\eta_1^3(U) = 5$ and $\eta_1^3(D) = 6$. We define the expected play by Bayesian updating so

$$\mu_i^t(a) = \frac{\eta_i^t(a)}{\sum_{a \in A} \eta_i^t(a)} \quad (3.19)$$

This equation represents the i player's estimate of opponent's strategy at time t as the empirical frequency distribution of past plays of each action a . Player i chooses an action to maximize his payoff according to

$$a_i^t = \arg \max_{a \in A_i} g_i(a, \mu_i^t) \quad (3.20)$$

where g_i is the payoff matrix for the player i . So the player maximizes his payoff in each round based on the beliefs about opponent's strategy. In our example we get for player 1 and action U

$$\mu_1^3(U) = \frac{5}{11} = 0.455$$

and for action D

$$\mu_1^3(D) = \frac{6}{11} = 0.545$$

so the attacker choses

$$a_1^3 = \arg \max_{a \in A_1} \{(3 * 0.455), (4 * 0.545 + 1 * 0.545)\} = \arg \max_{a \in A_1} \{(1.365), (2.725)\} = D \quad (3.21)$$

So the player 1 plays action D based on the opponent previous plays. An interesting feature of fictitious play is that the players do not have to know the opponent's payoff matrix. All the player's beliefs come from what the opponent played so far.

Chapter 4

Game Strategies

In this section we describe the properties of our game model for further experiments. In Section 2.4 there is a basic description of analyzed security game.

4.1 Player types

Our game model is a version of multi-armed bandit problem on the defender side and a version of a fictitious play on side of the attacker. We assume several types of defender strategies, which consist of online learning algorithms, game-theoretic strategies and combinations of them. The goal of the proposed model is to reflect real-world applications to border patrol and to show features of several different approaches with its drawbacks. We also point out the limitations of our model and propose possible extensions of the model. We focus on a repeated game model so we assume 1000 rounds, which can be imagined as 3 years of using this game model in border patrolling where each round represents 1 day. We also investigate some longer time periods to analyze the convergence behavior. We assume that all zones are identical for the defender and he gets payoff 1 if we apprehend the attacker and 0 otherwise.

4.1.1 Defender types

In our model we use these types of defenders:

- UCB algorithm

The defender uses UCB algorithm strategy to decide which zone to observe. This is a deterministic algorithm and is more described in Section 3.4.

- EXP3 algorithm

The defender uses EXP3 algorithm, which is suitable against adversarial opponents. This algorithm is stochastic using mixed probability strategy vector. This algorithm is described in 3.4.

- D-UCB algorithm

D-UCB algorithm is a version of UCB algorithm more suitable for non-stationary environments. It discounts the history of past rewards for computing the expected reward for each zone.

- SW-UCB algorithm

The defender uses for his strategy SW-UCB algorithm, which uses a sliding window of history of past rewards. This strategy is suitable for non-stationary environments.

- Stackelberg equilibrium strategy

This strategy comes from Stackelberg Security Game model. It assumes that the defender knows the exact attacker payoff matrix and therefore can compute the SSE. This strategy is the optimal stable strategy.

- Stackelberg equilibrium with an error strategy

This strategy assumes more realistic case where the defender does not know the exact attacker payoff matrix but at least knows an estimate of it. We assume that the defender estimate of the attacker strategy has an error. Based on this imprecise information the defender computes the SSE strategy. This strategy approach is further described in Section 4.3.2.

- Combined algorithms

We propose 4 combined algorithms of learning algorithm EXP3 and game-theoretic solution SSE. We describe COMB1, COMB2, COMB3 and COMB4 further in the text in Section 4.4.

- Combinatorial EXP3

We use combinatorial version of EXP3 described in Section 3.4.4. For COMB algorithms we use this combinatorial EXP3 and SSE strategy vector.

4.1.2 Attacker types

In our model we have several attacker types which differ in how adversary, realistic, adapting they are.

- Random fixed attacker

Random attacker's mixed strategy is a fixed probability distribution over all pure strategies and over all the rounds of the game. This attacker is very simplistic and easily exploitable. In reality this attacker corresponds to an attacker who does not remember any past history and thus is oblivious. It can also represent a group of attackers who do not cooperate. The probability distribution represents some explicit conditions of the actions. That is why some actions are more favorable than the others.

- Random fixed with changes attacker

This attacker strategy is similar as the previous one but the strategy vector randomly changes every given period. If we assume 1000 rounds per game we set changes every 200 rounds. So after 200 rounds the probability distribution vector changes randomly and stays fixed.

- Adversarial attacker

Adversarial attacker is based on a version of a fictitious play described in Section 3.5. Fictitious play is one of the basic learning rules and is very realistic in security domain.

It is a belief-based system, which means that the player forms beliefs about opponent strategy and behave rationally with respect to these beliefs. The adversarial attacker converges to Nash equilibrium under specific circumstances as described in Section 3.5. We describe our model as a Stackelberg game model so we break ties in favor to the defender as described in Section 2.3. The ties are quite rare since the attacker only observes the patrol history and the estimates are not exact. The attacker has a zone preference vector v_i^Ψ , which describes how likely it is for the attacker to cross the particular zone i . This vector can represent some explicit conditions in terrain how easy/difficult is to cross a zone. The attacker uses a fixed penalty of being caught P^Ψ which is 0.5 and computes his reward for a zone i according to

$$x_i^\Psi = (v_i^\Psi - P^\Psi * c_i^\Psi) \quad (4.1)$$

where c_i^Ψ is the probability of being caught for the attacker computed from patrol history, which is the estimation of defender strategy. So the attacker chooses action (zone) i such that

$$\arg \max_i x_i^\Psi \quad (4.2)$$

The attacker strategy is well showed on example in Section 2.4. This strategy is deterministic because the attacker always chooses the best response from his strategy vector x_i^Ψ .

- Adversarial with changes attacker

This strategy is very similar to the one before but the attacker zone preference vector v_i^Ψ changes every given period. For a game with 1000 rounds we change the vector every 200 rounds. This attacker strategy is not much realistic, because the attacker zone preferences do not change abruptly in real world, but it is a good tool for testing the defender algorithms stability against varying opponents.

- Nash equilibrium attacker

Nash attacker plays the Nash equilibrium. He has an exact knowledge about the defender's payoff. The attacker uses a mixed strategy Nash equilibrium vector.

4.2 Learning Algorithms Properties

4.2.1 UCB synchronization

We observe a strange behavior of UCB algorithm against the adversarial attacker. The total apprehension rate is higher than by playing Strong Stackelberg Equilibrium (SSE) strategy as shown in Figure 5.2a. This is caused by a deterministic nature of both strategies. The UCB algorithm is able to exploit the attacker and play very efficiently.

In the Table 4.2 there is an example of 10 rounds of a simple two-player game. This simple game differs to our game model used in this thesis only by number of zones. We can see how the defender and the attacker strategy develop. The players always choose a maximum of the expected reward out of those two zones. The maximum is written in bold, which represents

the chosen zone. Both players use vectors of *mean reward* and *patrol history*. In the first column there is a number of the round; if written in bold there was an apprehension of the attacker (the defender chose the same zone as the attacker).

Game model:

number of zones $n = 2$

number of rounds $T = 100$

zone preference for the attacker $v^\Psi = [0.51, 0.49]$

penalty for the attacker when being caught $P^\Psi = 0.5$

SSE strategy for the defender $\delta_\Theta = [0.52, 0.48]$

expected payoff playing SSE strategy is 52

payoff playing UCB strategy is 74

	1	2
1	1,0.01	0,0.49
2	0,0.51	1,-0.01

Table 4.1: Payoff matrix of UCB synchronization game model

$$a_t^\Theta = \arg \max_i \bar{x}_i + \sqrt{\frac{2 \ln n}{n_i}} \quad (4.3)$$

The defender plays UCB strategy (Equation 4.3), where x_i is the mean reward for zone i , n is the total number of rounds and n_i is the number of rounds when the defender played zone i . UCB algorithm is described in Section 3.4.

For the attacker we use the adversarial attacker strategy described in Section 4.1.2. Patrol history is a number of observations of a zone by the defender until the current round.

t	patrol h_i		mean reward x_i		defender strat a_t^Θ		attacker strat a_t^Ψ	
	1	2	1	2	1	2	1	2
30	16	14	0.875	0.857	1.6582	1.6944	0.243	0.257
31	16	15	0.875	0.867	1.6619	1.6794	0.252	0.248
32	16	16	0.875	0.813	1.6656	1.6031	0.260	0.240
33	17	16	0.882	0.813	1.6527	1.6066	0.252	0.248
34	18	16	0.889	0.813	1.6407	1.6100	0.245	0.255
35	19	16	0.842	0.813	1.5769	1.6132	0.239	0.261
36	19	17	0.842	0.824	1.5798	1.6034	0.246	0.254
37	19	18	0.842	0.833	1.5826	1.5941	0.253	0.247
38	19	19	0.842	0.789	1.5854	1.5327	0.260	0.240
39	20	19	0.850	0.789	1.5770	1.5354	0.254	0.246
40	21	19	0.857	0.789	1.5691	1.5379	0.248	0.253

Table 4.2: Game example of UCB synchronization

In Table 4.2 we can see a part of the game, which helps to understand the UCB synchronization with the adversarial attacker. We can observe 10 rounds of the game; rounds

30 – 40. In the table there is the patrol history, mean rewards for both zones and the strategy vectors of both players from which the players choose the zone which gives the maximum of argument. Values of these vectors in bold are the chosen zones by the players. Numbers of rounds in bold are those rounds where the defender chose the same zone as the attacker, therefore the defender apprehended the attacker.

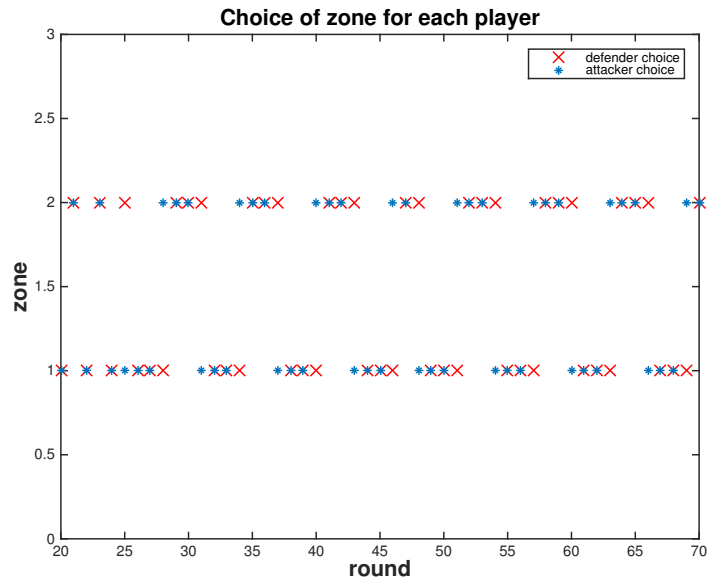


Figure 4.1: Chosen zones for the players in UCB synchronization

In Figure 4.1 we can see small example of such synchronization, there are zones chosen by each of the players. If both symbols are at the same zone then the attacker was apprehended by the defender.

We stated that expected payoff for playing Stackelberg equilibrium is 52 for the described game and playing the UCB learning algorithm we get payoff equals to 74. By comparing these two strategies we can conclude that the attacker is exploited by the defender. This is caused by the deterministic nature of both algorithms and their ability to learn the opponent strategy. The exploitation is not a desirable property in the security model, because such behavior would probably lead to one player changing the strategy. In this case the attacker would most probably change his strategy, because he would get caught in 74 cases out of 100 tries of crossing the border. Therefore in our model we will focus on algorithms with some randomness (stochastic algorithms).

4.2.2 Combinatorial EXP3 implementation

4.2.2.1 Projection heuristic

The implementation of the COMB-EXP-1 algorithm is straightforward apart from the I-projection using KL-divergence. As stated in paper [11] the I-projection is defined as such a distribution p^* onto a closed convex set Ξ of distributions, which has the minimal KL-

divergence from vector q .

$$KL(p^*, q) = \min_{p \in \Xi} KL(p, q) \quad (4.4)$$

where KL-divergence is defined

$$KL(q, p) = \sum_{i \in [d]} q(i) \log \frac{q(i)}{p(i)} \quad (4.5)$$

So the sought distribution p^* must be a probability distribution vector, thus it sums up to 1. Furthermore to guarantee that the step 1 in Algorithm 5 in Section 3.4.4 for computing the distribution p has always a solution the sought projection distribution p^* must hold the property that no value in the vector is greater than $1/k$ where k is the number of resources we have. There is no known general algorithm to us for computing such projection and it is believed to be a hard open problem [10]. Thus, we propose a heuristic algorithm and show that it is good enough by comparing it to other methods.

In our heuristic we decrease all values greater than $1/k$ to $1/k$ and normalize all other values in the vector to $(1 - a/k)$, where a is the number of values in the original vector greater than $1/k$. We show an example of performed experiments of testing our heuristic H_1 where there is a vector with one value greater than maximal possible value:

- $q = [0.6, 0.3, 0.05], k = 2$
- we firstly normalize the vector so $q' = [0.6316, 0.3158, 0.0526]$
- we decrease the values which are greater than $k/2 = 0.5$
- $\bar{q} = [0.5, 0.3158, 0.0526]$
- and normalize other values to $1 - a/k = 0.5$ because we have only one value greater than $1/k$, thus $a = 1$.
- we get the projection $p = [0.5, 0.4286, 0.0714]$
- KL-divergence for p and q is $KL(p, q) = 0.0359$

we can compare this heuristic to another heuristic H_2 where we redistribute the difference value from value in a vector greater than $1/k$ among other values, such as

- $q = [0.6, 0.3, 0.05], k = 2$
- we firstly normalize the vector so $q' = [0.6316, 0.3158, 0.0526]$
- we decrease the values which are greater than $k/2 = 0.5$
- $\bar{q} = [0.5, 0.3158, 0.0526]$ and we get the difference value $x = q'(1) - \bar{q}(1) = 0.1316$
- now we add the difference value equally to all other values in the vector
- we add $x/2 = 0.0658$ to the second and the third value so we get the projection vector $\bar{p} = [0.5, 0.3816, 0.1184]$
- KL-divergence for p and q is $KL(p, q) = 0.0514$

We can see that the proposed heuristic algorithm H_1 gives us lower KL-divergence, thus is better. We also show KL-divergence values for modified projection vector:

- initial vector $q = [0.6, 0.3, 0.05]$ and $k = 2$
- H_1 heuristic give projection $p = [0.5, 0.4286, 0.0714]$ with $KL(p, q) = 0.0359$
- we shift the values in the vector to multiple directions by some constant $\xi = 0.01$ to see how the KL-divergence changes
- 1. $p_2 = [0.5, 0.4186, 0.0814]$ we get $KL(p_2, q) = 0.0367$
- 2. $p_3 = [0.5, 0.4386, 0.0614]$ we get $KL(p_3, q) = 0.0367$
- 3. $p_4 = [0.49, 0.4336, 0.0764]$ we get $KL(p_4, q) = 0.0416$

We showed an example of our heuristic and the comparison to other methods. We test the heuristic in such manner on large number of experiments. We generate a random vector of size 8, which is the basic number of zones used in our game model and compare the heuristic H_1 and H_2 . In 10000 experiments with randomly generated vectors and with different number of resources k there is H_1 heuristic always better than H_2 . Then we run experiments with shifted values to different directions. In 10000 experiments for randomly generated vectors of size 8 and different numbers of resources and different values of decrease/increase constant ξ in range $[0.01, 0.00001]$ we get in less than 1% of the cases better result for this shifting method than for using H_1 heuristic. Comparing different approaches to compute the projection using KL-divergence we can see that our heuristic H_1 gives us mostly the lowest value of KL-divergence and therefore we can state that the proposed heuristic is reasonably good and we can use it for our experiments.

4.3 Game-theoretic Strategies

In our game model we analyze game-theoretic solution. Since our game is a version of Stackelberg security game we focus on Stackelberg equilibrium (SE) mixed strategy. To be able to find the exact game equilibrium for the defender, he needs to know the exact attacker payoff. This is very problematic in real applications because the defender might know an estimate of attacker payoffs but usually not the exact values.

4.3.1 Stackelberg equilibrium strategy

We assume that the defender knows the exact attacker payoff matrix and therefore the defender is able to compute precise Stackelberg equilibrium. We compare our proposed algorithms to SE strategy, because it is the optimal strategy.

4.3.2 Estimated Stackelberg equilibrium strategy

In real applications we can assume that the defender does not have the exact attacker payoff, thus the defender is not able to compute the exact Stackelberg equilibrium strategy. We analyze the impact of error in attacker payoff estimate on the performance of defender playing the estimated SSE in Section 5.2.1. We assume a random error ϵ in each value of attacker zone

preference vector (described in Section 2.4). The added error to each value can be positive or negative, so we add a random number from range of size ϵ with mean value 0. We use this estimated vector to compute the Stackelberg equilibrium.

4.3.3 Motivating Stackelberg equilibrium strategy

Our model is a version of Stackelberg security game as described in Section 2.3. However in our model the follower(attacker) does not know the exact leader(defender) strategy, the attacker has only an estimate of this strategy which is by its nature imprecise. The attacker computes this estimate from patrol history, which is the number of defender patrols in each zone. So when the attacker chooses a zone to attack (best-response action), which has the highest expected payoff, he does the decision based on this imprecise estimate of the defender strategy. Due to this fact the attacker does not always play the optimal strategy for him, therefore the defender does not get the expected payoff for Stackelberg game in case of playing the Stackelberg equilibrium. We introduce a simple modification to the defender Stackelberg equilibrium strategy, which motivates the attacker to play the desired action in spite of having imprecise defender strategy. To increase the total payoff and get closer value to the expected SE payoff we present a method to motivate the attacker to choose the zone we want him to choose. We decrease the highest value in SSE strategy vector by a constant value and normalize the probability vector. This change in the strategy vector motivates the attacker to choose more likely the particular zone, which he would normally choose in Stackelberg game playing strong Stackelberg equilibrium where the attacker breaks ties in favor to the defender. We perform several experiments to test this concept in Section 5.2.2.

4.4 Combined Algorithms

In this section we propose 4 combined algorithms, which use EXP3 online learning algorithm and game-theoretic solution. We present COMB1, COMB2, COMB3 and COMB4 algorithm.

4.4.1 Combination algorithm 1

In Section 3.4.2 we described the EXP3 algorithm. It computes the expected reward for each zone. In that section in Equation 3.5 we can see that EXP3 computes the expected payoff for each action (zone) by difference of sums of past payoffs divided by probability of playing this action in that past round. We can initialize EXP3 algorithm by adding values to this expected payoff vector. If we assume that optimally the attacker strategy converges to Nash equilibrium, we can initialize defender EXP3 algorithm by the attacker Nash equilibrium. We compute the Nash equilibrium for both players using the payoff matrix. We assume that the equilibrium is not precise since the attacker payoff is estimated with an error. Then we use the attacker Nash equilibrium for the EXP3 learning algorithm initialization. This vector estimates the expected payoff for each zone. We use a parameter τ which represents how certain we are of the estimated zone preference vector, thus it means how precise is the Nash equilibrium; the more certain we are the higher τ parameter will be. One can think of τ as a number of virtual rounds we played before start of the game. So we compute the initialization vector

$$r_i = N_i * \tau \quad (4.6)$$

where N_i is the Nash equilibrium for zone i . Then we run the standard EXP3 learning algorithm with this r initialization vector.

4.4.2 Combination algorithm 2

We compute the Nash equilibrium based on estimated attacker's zone preference. The idea for the defender is to play its Nash equilibrium strategy with an extra exploration 10% first T rounds and then to play EXP3 learning algorithm, which uses the information about expected payoffs for each zone gathered from the previous rounds. For finding the point where to switch from first stage to the second we compute EXP3 payoff virtually during the play of estimated Nash equilibria. Virtual EXP3 payoff is computed so that it gives higher payoff for a strategy with higher probability of visiting a particular zone. So if the probability of EXP3 of visiting a particular zone with positive payoff is higher than probability in Nash equilibrium vector, we get relatively higher payoff for EXP3 than for NE strategy. In this manner we prioritize a strategy, which gives us higher payoff. If the defender is in the same zone as the attacker, the defender gets payoff 1 and virtual EXP3 payoff is

$$p_{EXP3}^d(t) = \frac{e_i^t}{n_i^t} * 1 \quad (4.7)$$

or if the defender is not in the same zone as the attacker; the defender gets payoff 0 and virtual EXP3 payoff is

$$p_{EXP3}^d(t) = \frac{e_i^t}{n_i^t} * 0 \quad (4.8)$$

where e_i^t is the probability of playing zone i in round t playing EXP3 and n_i^t is the probability of playing zone i in round t by the estimated Nash equilibrium strategy.

We compute the total payoff for both strategies concepts as the sum over all the rounds played so far. The algorithm switches to EXP3 learning algorithm if

$$P_{EXP3} > P_{NE} \quad (4.9)$$

where P_{EXP3} is the total payoff of virtually playing EXP3 learning algorithm. And P_{NE} is the total payoff (number of apprehensions) of playing estimated Nash equilibrium with an exploration rate 10%.

Once we switch to EXP3 algorithm, it uses the information about rewards from previous rounds. We focus on this basic setting but there can be used a stabilization parameter ζ which filter some early behavior and thus we can assume the strategy switching when

$$P_{EXP3} - \zeta > P_{NE} \quad (4.10)$$

4.4.3 Combination algorithm 3

We propose a similar concept as in the previous combined algorithm but now we switch between two strategies according to the highest total payoff. For the strategy we are playing we store the total actual payoff and for the other strategy we compute the payoff as we virtually play it. So analogically for virtually playing NE strategy we get payoff

$$p_{NE}^d(t) = \frac{n_i^t}{e_i^t} * 1 \quad (4.11)$$

or if there is no apprehension

$$p_{NE}^d(t) = \frac{n_i^t}{e_i^t} * 0 \quad (4.12)$$

So we play the estimated Nash equilibria with exploration if

$$P_{EXP3} < P_{NE} \quad (4.13)$$

or we play the EXP3 if

$$P_{EXP3} > P_{NE} \quad (4.14)$$

The EXP3 algorithm uses the expected payoff vector from all previously played rounds including those rounds when the defender played the NE strategy with exploration.

4.4.4 Combination algorithm 4

The defender has several estimated Nash equilibria strategy vectors computed from the attacker zone preference vector with a random error of constant size ϵ as described in Section 4.3.2. We start playing with one of the strategy and parallelly compute the expected payoffs for other estimated Nash strategies and for EXP3 learning algorithm. We switch between these strategies according to the highest payoff. In our model we did experiments with 3 estimated Nash equilibria (NE). These NE strategies are called experts. So we use an expert strategy model.

4.5 Combinatorial Combined Algorithms

We use Stackelberg equilibrium for multiple resources and combinatorial EXP3 learning algorithm. The combinatorial COMB algorithms are analogical to COMB algorithms for the standard non-combinatorial case, however there are some modifications needed as described in the following sections.

4.5.1 Combinatorial COMB1

In combinatorial COMB1 algorithm we use the estimated Stackelberg equilibrium (SE) strategy to initialize EXP3 algorithm. At the beginning of the game we set the defender strategy as stated in the following equation and normalize the vector.

$$a_1^\Theta = \tau * SE + (1 - \tau) * U \quad (4.15)$$

where τ is the parameter which sets how confident we are about the Stackelberg equilibrium strategy. The basic setting is $\tau = 0.9$. SE is the Stackelberg equilibrium strategy vector and U is the uniform probability vector. From the Section 3.4.4, where there is combinatorial EXP3 description, we can see that mean reward vector X is computed only from the previous-round strategy vector. Therefore we can initialize EXP3 algorithm at the beginning by SE strategy and the uniform probability vector. As we can see in the combinatorial EXP3 algorithm description, the initial state of the algorithm is the uniform vector. So by setting $\tau = 0$ we do not initialize EXP3 algorithm at all and play the standard combinatorial online learning algorithm. After initializing the online learning algorithm we continue playing standard combinatorial EXP3.

4.5.2 Combinatorial COMB2

Combinatorial COMB2 algorithm is analogical to standard COMB2 algorithm. We use the estimated Stackelberg equilibrium strategy for multiple resources with 10% extra exploration and combinatorial EXP3 algorithm. We start with SE strategy and compute virtually expected payoff for playing EXP3. Once the virtual EXP3 payoff becomes greater than actual payoff by playing SE with extra exploration we switch to EXP3 algorithm using SE strategy vector as the initialization for the combinatorial EXP3.

4.5.3 Combinatorial COMB3

Analogically to non-combinatorial COMB algorithms, combinatorial COMB3 algorithm is a generalization of previous combinatorial COMB2 algorithm. In this COMB3 algorithm we enable the switching between the two strategies arbitrary according to the highest payoff. We compute the virtual SE strategy payoff while playing EXP3 algorithm and vice versa.

4.5.4 Combinatorial COMB4

For this combined algorithm we have several estimated Stackelberg equilibria strategy computed from the zone preference vector with different random errors of constant size ϵ . In our experiments we work with 3 SE strategies or in other words with 3 experts. Combinatorial COMB4 is analogical to standard COMB4.

Chapter 5

Experiments

In this section we do several experiments to show behavior and performance of the learning algorithms. We also focus on game-theoretic strategies and analyze their usability in our model. We test the proposed combined algorithms of the two approaches. At the end we investigate the combinatorial case and do several experiments to describe the behavior of the model. Each experiment with particular setting is run multiple times to get statistically stable result, each time with newly randomly generated vectors for the selected attacker type. So against the random fixed attacker in each experiment there is a new random vector and against the adversarial attacker there is a new random zone preference vector for each experiment. Each game in our game model has 1000 rounds. We compare the algorithms by apprehension rates, which is a percentage of apprehended attackers out of all attackers who tried to cross the border. The figures visualize the apprehension rate by cumulative averages from the beginning. So the actual cumulative performance in every round is its derivation. In Figures 5.6 there are examples of moving-averages visualization, where one can get better understanding of the graph representation. For each experiment we provide width of a 95% confidence interval, one can see such interval visualized in Figure 5.5b.

5.1 Experiments with Learning Algorithms

We analyze performance of the learning algorithms proposed in Section 3.4. We begin with parameter tuning. We show several experiments to find the optimal values of the parameters for our game model. We also observe convergence behavior and comparison among the proposed learning algorithms. We run every experiment 1000 times to get statistically relevant results and we provide 95% confidence intervals. For the parameter tuning we assume 1 attacker tries to cross the border in each round. Our game model is described in Section 2.4.

5.1.1 Parameter tuning

Learning algorithms have parameters, which determine the algorithm performance. These parameters are either dependent on other features of the game model setting or they can be constants. As we described in Section 3.4 on learning algorithms, there are usually some parameters, which can be tuned to adapt the learning algorithm to the game model. In this section we do several experiments to tune the algorithms to perform as good as possible in our proposed game model. By tuning the parameters we also show the features and behavior

of the proposed algorithms so that one can better understand them.

5.1.1.1 EXP3 exploration versus exploitation trade-off

In Section 3.4.2 we can see that algorithm EXP3 has the exploration parameter γ . Widely used general setting for this parameter γ is 0.1, which means 10% of exploration. We make several experiments to find the best $\gamma \in (0, 1)$ setting for our model.

EXP3	random		random with changes	
γ value	mean	confidence	mean	confidence
0	12.52%	$\pm 0.06\%$	12.50%	$\pm 0.06\%$
0.1	15.29%	$\pm 0.14\%$	13.00%	$\pm 0.07\%$
0.2	16.43%	$\pm 0.17\%$	13.48%	$\pm 0.09\%$
0.3	16.78%	$\pm 0.18\%$	13.70%	$\pm 0.09\%$
0.4	16.73%	$\pm 0.17\%$	13.73%	$\pm 0.09\%$
0.5	16.06%	$\pm 0.15\%$	13.68%	$\pm 0.08\%$
0.6	15.54%	$\pm 0.12\%$	13.49%	$\pm 0.08\%$
0.7	14.78%	$\pm 0.10\%$	13.30%	$\pm 0.08\%$
0.8	14.07%	$\pm 0.09\%$	13.06%	$\pm 0.07\%$
0.9	13.30%	$\pm 0.07\%$	12.74%	$\pm 0.07\%$
1	12.52%	$\pm 0.07\%$	12.51%	$\pm 0.06\%$

Table 5.1: Tuning parameter γ of EXP3 algorithm against random attacker

EXP3	adversarial		adversarial with changes	
γ value	mean	confidence	mean	confidence
0	12.51%	$\pm 0.06\%$	12.43%	$\pm 0.07\%$
0.1	19.92%	$\pm 0.28\%$	16.97%	$\pm 0.17\%$
0.2	19.81%	$\pm 0.28\%$	18.56%	$\pm 0.22\%$
0.3	19.75%	$\pm 0.28\%$	19.08%	$\pm 0.23\%$
0.4	19.97%	$\pm 0.28\%$	18.74%	$\pm 0.21\%$
0.5	19.85%	$\pm 0.28\%$	18.12%	$\pm 0.19\%$
0.6	19.63%	$\pm 0.27\%$	17.57%	$\pm 0.17\%$
0.7	19.42%	$\pm 0.24\%$	16.42%	$\pm 0.14\%$
0.8	18.60%	$\pm 0.21\%$	15.47%	$\pm 0.11\%$
0.9	17.18%	$\pm 0.14\%$	14.29%	$\pm 0.09\%$
1	12.47%	$\pm 0.06\%$	12.47%	$\pm 0.06\%$

Table 5.2: Tuning parameter γ of EXP3 algorithm against adversarial attacker

In Table 5.1 we can see apprehension rates for different values of parameter γ . In the columns we have random fixed attacker types as described in Section 4.1.2. Random fixed attacker with changes modifies randomly his mixed strategy every 200 rounds. In bold there are the highest apprehension rates for each attacker. There is also the 95% confidence interval provided. We can see that for $\gamma \in (0.3, 0.4)$ we get the highest apprehension rate against the random fixed attacker with respect to the confidence interval. Against the random fixed attacker with changes we get the best performance for $\gamma \in (0.3, 0.5)$. The uniform defender

strategy gives apprehension rate 12.5% because we choose 1 zone out of 8 zones so $1/8 = 0.125$. One can observe that for $\gamma = 0$ or $\gamma = 1$ the performance is the same as playing the uniform strategy. For $\gamma = 1$ we only explore the zone space so in every round we play uniform strategy. As stated and described in Section 3.4.2 we use in algorithm EXP3 parameter $\eta = \frac{\gamma}{K}$, where K is the number of zones. For $\gamma = 0$ we get $\eta = 0$ and therefore we also play only uniform strategy in every round. This comes from the nature of the algorithm and it holds against multiple attacker types.

In Table 5.2 we tuned the parameter γ against the adversarial attacker types described in Section 4.1.2. We can state that for $\gamma \in (0.1, 0.6)$ we get the highest apprehension rate in 95% confidence interval. Against the adversarial attacker with changes the highest apprehension rates are for $\gamma \in (0.3, 0.4)$. We can also see that for $\gamma = 0$ or $\gamma = 1$ we get the results as playing uniform strategy over all the zones.

In our model we mainly focus on the case against the adversarial attacker so we use $\gamma = 0.2$ for further experiments, which gives one of the highest apprehension rates with respect to the confidence bound against adversarial attacker.

5.1.1.2 Discounted UCB parameters

For discounted UCB (D-UCB) we tune two parameters. Firstly the parameter B , which controls the exploration part of the algorithm and secondly the discount factor γ as described in Section 3.4. For parameter B tuning we set the discount factor $\gamma = 0.8$.

D-UCB, $\gamma = 0.8$	random		random with changes	
B value	mean	confidence	mean	confidence
0	13.48%	$\pm 0.45\%$	12.66%	$\pm 0.21\%$
0.01	15.82%	$\pm 0.15\%$	15.61%	$\pm 0.10\%$
0.03	14.77%	$\pm 0.12\%$	14.78%	$\pm 0.09\%$
0.05	14.40%	$\pm 0.11\%$	14.35%	$\pm 0.08\%$
0.07	14.04%	$\pm 0.10\%$	14.10%	$\pm 0.08\%$
0.1	13.85%	$\pm 0.09\%$	13.80%	$\pm 0.07\%$
0.2	13.41%	$\pm 0.08\%$	13.36%	$\pm 0.07\%$
0.3	13.10%	$\pm 0.07\%$	13.07%	$\pm 0.07\%$
0.5	12.89%	$\pm 0.07\%$	12.90%	$\pm 0.07\%$

Table 5.3: Tuning parameter B of D-UCB algorithm against random attacker

In Table 5.3 we tune the parameter B against the random fixed attacker types. The best performance is for $B = 0.01$ with respect to the confidence intervals. We are better off with lower levels of exploration because the defender plays against attacker who plays a static mixed strategy without any learning, so the defender does not need to explore much.

D-UCB, $\gamma = 0.8$	adversarial		adversarial with changes	
B value	mean	confidence	mean	confidence
0	0.21%	$\pm 0.01\%$	0.33%	$\pm 0.06\%$
0.01	26.57%	$\pm 0.28\%$	42.96%	$\pm 0.32\%$
0.03	27.97%	$\pm 0.26\%$	42.51%	$\pm 0.29\%$
0.05	28.90%	$\pm 0.23\%$	42.33%	$\pm 0.28\%$
0.07	29.69%	$\pm 0.23\%$	41.83%	$\pm 0.26\%$
0.1	30.50%	$\pm 0.20\%$	40.74%	$\pm 0.22\%$
0.2	32.42%	$\pm 0.17\%$	38.77%	$\pm 0.14\%$
0.3	31.28%	$\pm 0.17\%$	34.54%	$\pm 0.08\%$
0.5	31.83%	$\pm 0.27\%$	27.81%	$\pm 0.09\%$

Table 5.4: Tuning parameter B of D-UCB algorithm against adversarial attacker

In Table 5.4 there is the parameter tuning against adversarial attacker types. We can observe that for $B = 0$ the attacker exploits the defender and almost avoid being caught. This is caused by zero exploration, thus the defender has no way how to learn about the busier zones. Against the attacker with fixed zone preference vector we get the best performance for $B = 0.2$ and against the attacker with changes in his zone preference vector the highest apprehension rate is for $B \in (0.01, 0.03)$. One can see that against adversarial attacker one is better off with more exploration, which is caused by the attacker learning process. The defender with more exploration confuses the attacker more and thus it is harder for the attacker to learn the defender strategy, therefore the defender gets high apprehension rate. However against the adversarial attacker with changes we are better off with less exploration. The defender adapts his strategy to these changes in attacker strategy and makes the attacker difficult to learn the defender strategy even with low exploration. Higher exploration makes the defender to learn poorly the attacker strategy and therefore the defender gets lower apprehension rate.

D-UCB, $B = 0.3$	random		random with changes	
γ value	mean	confidence	mean	confidence
0	12.83%	$\pm 0.44\%$	12.74%	$\pm 0.21\%$
0.1	12.53%	$\pm 0.06\%$	12.49%	$\pm 0.06\%$
0.2	12.52%	$\pm 0.06\%$	12.54%	$\pm 0.06\%$
0.3	12.53%	$\pm 0.06\%$	12.49%	$\pm 0.06\%$
0.4	12.47%	$\pm 0.07\%$	12.52%	$\pm 0.06\%$
0.5	12.57%	$\pm 0.06\%$	12.56%	$\pm 0.06\%$
0.6	12.59%	$\pm 0.07\%$	12.62%	$\pm 0.06\%$
0.7	12.80%	$\pm 0.07\%$	12.84%	$\pm 0.07\%$
0.8	13.04%	$\pm 0.08\%$	13.14%	$\pm 0.07\%$
0.9	13.73%	$\pm 0.27\%$	13.66%	$\pm 0.13\%$
1	19.70%	$\pm 0.07\%$	17.46%	$\pm 0.06\%$

Table 5.5: Tuning parameter γ of D-UCB algorithm against random attacker

We tune the parameter γ in Table 5.5 for $B = 0.3$. We can see that for discount factor $\gamma = 1$ there are the highest apprehension rates against random attackers. This means the

D-UCB algorithm is not suitable against random attackers, which support the fact that D-UCB is recommended for non-stationary environments. For $\gamma = 1$ there is no discount of the past rewards for computing the expected reward for each action (zone) so it boils down to the standard UCB with some multiplication constant B .

D-UCB, $B = 0.3$	adversarial		adversarial with changes	
γ value	mean	confidence	mean	confidence
0	0.21%	$\pm 0.01\%$	0.21%	$\pm 0.01\%$
0.1	13.08%	$\pm 0.09\%$	13.05%	$\pm 0.06\%$
0.2	13.06%	$\pm 0.09\%$	13.05%	$\pm 0.05\%$
0.3	13.05%	$\pm 0.09\%$	13.07%	$\pm 0.06\%$
0.4	13.08%	$\pm 0.09\%$	13.03%	$\pm 0.05\%$
0.5	16.93%	$\pm 0.28\%$	15.36%	$\pm 0.08\%$
0.6	22.26%	$\pm 0.39\%$	18.73%	$\pm 0.11\%$
0.7	30.71%	$\pm 0.35\%$	25.71%	$\pm 0.12\%$
0.8	31.31%	$\pm 0.17\%$	34.50%	$\pm 0.09\%$
0.9	32.08%	$\pm 0.22\%$	41.90%	$\pm 0.22\%$
1	23.85%	$\pm 0.25\%$	30.15%	$\pm 0.25\%$

Table 5.6: Tuning parameter γ of D-UCB algorithm against adversarial attacker

Against the adversarial attacker types we are better off with the discount factor $\gamma = 0.9$ against both adversarial attacker types. We can also notice that for $\gamma = 0$ the defender does not apprehend almost any attacker; this is caused by the nature of the algorithm where we compute neither the mean reward nor the exploration. For $\gamma = 1$ there is a drop in performance so the discounting of past rewards helps the performance. We can also observe that the defender gets higher apprehension rate against the attacker with changes, which is not intuitive. This is caused by the fact that the attacker does not have time to learn effectively the defender strategy because the defender strategy adapts to the changes in attacker strategy faster.

The advantage of D-UCB appears against more intelligent attackers who can adapt their strategies as we can see from the experiments above. For $\gamma = 1$ D-UCB boils down to standard UCB multiplied by parameter B . For standard UCB holds $B = \sqrt{2}$.

5.1.1.3 Sliding-window UCB parameters

We tune two parameters for the Sliding-window UCB (SW-UCB). Firstly the parameter B which controls the exploration part of the algorithm and secondly the size of the sliding window τ as described in Section 3.4. For parameter B tuning we set $\tau = 300$.

SW-UCB, $\tau = 300$	random		random with changes	
B value	mean	confidence	mean	confidence
0	13.69%	$\pm 0.45\%$	12.72%	$\pm 0.21\%$
0.1	20.30%	$\pm 0.28\%$	16.63%	$\pm 0.13\%$
0.3	18.04%	$\pm 0.23\%$	17.22%	$\pm 0.13\%$
0.5	16.42%	$\pm 0.20\%$	16.21%	$\pm 0.11\%$
0.7	15.58%	$\pm 0.16\%$	15.38%	$\pm 0.10\%$
1	14.63%	$\pm 0.12\%$	14.61%	$\pm 0.09\%$
2	13.58%	$\pm 0.08\%$	13.54%	$\pm 0.07\%$
5	12.87%	$\pm 0.07\%$	12.98%	$\pm 0.07\%$
10	12.65%	$\pm 0.06\%$	12.71%	$\pm 0.07\%$

Table 5.7: Tuning parameter B of SW-UCB algorithm against random attacker

In Table 5.7 we tune the parameter against the random fixed attacker types. Against the random fixed attacker we get the best result for $B = 0.1$ and against the attacker with changes for $B = 0.3$. We can see that the defender is better off for lower exploration levels due to the fact that the attacker plays a static mixed strategy and does not learn and thus the defender does not need to explore that much and rather exploit the best zones.

SW-UCB, $\tau = 300$	adversarial		adversarial with changes	
B value	mean	confidence	mean	confidence
0	0.21%	$\pm 0.01\%$	0.22%	$\pm 0.02\%$
0.1	14.21%	$\pm 0.28\%$	16.70%	$\pm 0.32\%$
0.3	21.94%	$\pm 0.25\%$	33.15%	$\pm 0.38\%$
0.5	25.33%	$\pm 0.26\%$	39.90%	$\pm 0.40\%$
0.7	27.13%	$\pm 0.27\%$	42.33%	$\pm 0.38\%$
1	28.77%	$\pm 0.26\%$	43.60%	$\pm 0.36\%$
2	31.02%	$\pm 0.22\%$	41.77%	$\pm 0.27\%$
5	28.01%	$\pm 0.13\%$	28.29%	$\pm 0.09\%$
10	21.89%	$\pm 0.23\%$	20.09%	$\pm 0.10\%$

Table 5.8: Tuning parameter B of SW-UCB algorithm against adversarial attacker

Experiments with the parameter tuning against the adversarial attacker types are in Table 5.8. Against the adversarial attacker we get the highest apprehension rate for $B = 2$ and against the attacker with changes for $B = 1$. So one can conclude that one needs to explore more against more intelligent adversarial attackers who can learn the defender strategy. We can also observe that the defender gets higher apprehension rate against the attacker with changes than against the attacker without changes, this behavior is explained above for Table 5.6.

SW-UCB, $B = 1$	random		random with changes	
τ value	mean	confidence	mean	confidence
50	13.47%	$\pm 0.08\%$	13.41%	$\pm 0.07\%$
70	13.61%	$\pm 0.08\%$	13.57%	$\pm 0.07\%$
100	13.80%	$\pm 0.08\%$	13.73%	$\pm 0.07\%$
200	14.34%	$\pm 0.10\%$	14.30%	$\pm 0.08\%$
300	14.64%	$\pm 0.12\%$	14.59%	$\pm 0.09\%$
400	14.90%	$\pm 0.13\%$	14.88%	$\pm 0.09\%$
500	15.17%	$\pm 0.13\%$	15.06%	$\pm 0.10\%$
600	15.17%	$\pm 0.13\%$	15.08%	$\pm 0.10\%$
700	15.37%	$\pm 0.15\%$	15.14%	$\pm 0.11\%$
1000	16.15%	$\pm 0.18\%$	15.33%	$\pm 0.11\%$

Table 5.9: Tuning parameter τ of SW-UCB algorithm against random attacker

We observe the algorithm behavior for different values of the sliding window τ in Table 5.9. Against the random attacker types the best performance is for τ equals to the number of rounds played, which is basically the standard UCB multiplied by a constant B . For the case with changes $\tau \in (700, 1000)$ gives the highest apprehension rates. This supports the idea beyond SW-UCB algorithm that it is designed for non-stationary environments and thus does not perform that well against stationary attackers who cannot learn the defender strategy.

SW-UCB, $B = 1$	adversarial		adversarial with changes	
τ value	mean	confidence	mean	confidence
50	25.06%	$\pm 0.23\%$	35.69%	$\pm 0.26\%$
70	25.35%	$\pm 0.25\%$	37.35%	$\pm 0.27\%$
100	25.82%	$\pm 0.26\%$	39.75%	$\pm 0.31\%$
200	27.95%	$\pm 0.26\%$	42.97%	$\pm 0.34\%$
300	28.72%	$\pm 0.26\%$	43.89%	$\pm 0.37\%$
400	29.43%	$\pm 0.27\%$	43.51%	$\pm 0.35\%$
500	30.00%	$\pm 0.27\%$	43.74%	$\pm 0.36\%$
600	30.12%	$\pm 0.28\%$	43.74%	$\pm 0.35\%$
700	30.25%	$\pm 0.27\%$	43.31%	$\pm 0.35\%$
1000	31.07%	$\pm 0.27\%$	43.57%	$\pm 0.32\%$

Table 5.10: Tuning parameter τ of SW-UCB algorithm against adversarial attacker

Against adversarial attacker types in Table 5.10 we get the best performance for $\tau = 1000$ in case of the adversarial attacker, which means we get better results with the whole reward history. The highest apprehension rate against the attacker with changes is for $\tau \in (300, 1000)$. So we can observe that for varying environments we might be better off with shorter time windows, because after the attacker changes his strategy the defender does not need the whole reward history but only rewards after the change.

We showed properties of the proposed learning algorithms and their sensitivity to parameter setting. This can help to better understand the nature of studied learning algorithms.

5.1.2 Algorithm performance comparison

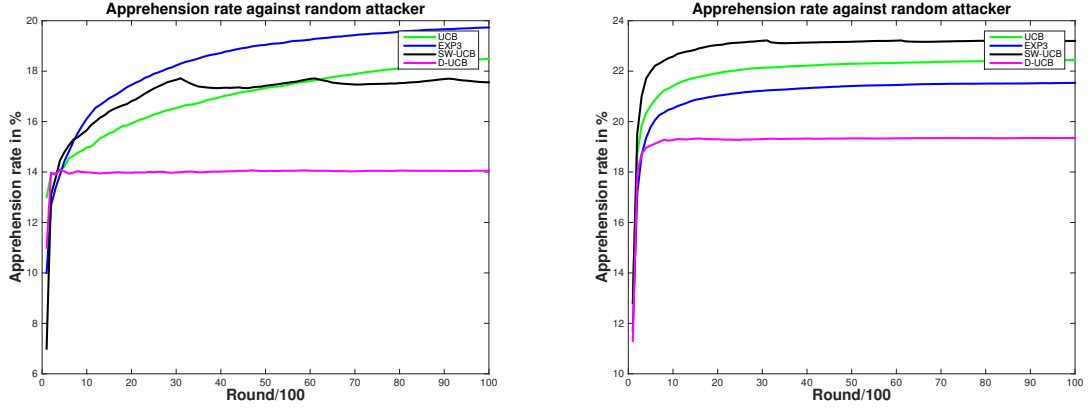
In this work we mainly focus on horizon of 1000 rounds per game and we do most of the analysis on this time window but it is interesting to look further because we can observe convergence behavior of proposed algorithms and compare them with each other. Furthermore we make sure the performance is stable and the attacker cannot exploit the defender in longer run. We study experiments with 10000 rounds run 100 times to get stable results. There are experiments with 4 learning algorithms against the random fixed attacker types. We also analyze a model with 10 attackers per round. If we have 10 attackers per round there is stronger information for the defender obtained in every round and the learning should be faster. These attackers are assigned to the zones by sampling from the strategy distribution vector multiple times. Against the adversarial attacker there are experiments with 4 learning algorithms and Strong Stackelberg Equilibrium (SSE) strategy for comparison. SSE is added to the experiments for the reference to further experiments with combined algorithms. The adversarial attacker is deterministic so the attacker does not have a probability strategy vector therefore we investigate only the case with one attacker per round since more attackers assigned to one zone have the same effect as multiplying the algorithm by some constant. On the x-axis there is a number of round and on y-axis there is the apprehension rate in %, which represents how many attackers were apprehended out of all attackers who tried to cross the border. We also provide 95% confidence interval, for each figure we state the maximal mean confidence interval. We do experiments with these parameters of learning algorithms:

EXP3 - $\gamma = 0.2$

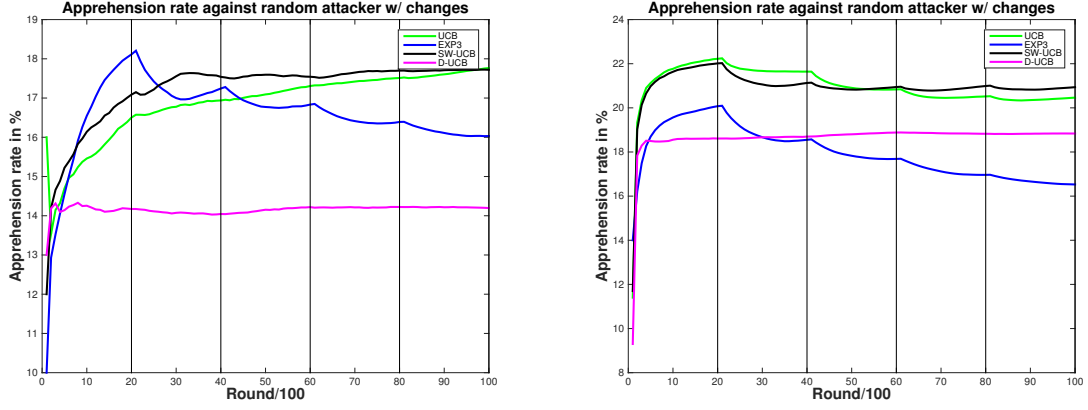
SW-UCB - $\tau = 3000, B = 1$

D-UCB - $\gamma = 0.9, B = 0.2$

This parameter setting is strongly inspired from the previous chapter, where we tuned these parameters. We focus on performance against adversarial attacker types. For SW-UCB we set $\tau = 3000$ because we extended the number of rounds 10 times and we investigate if this analogy holds with increasing number of rounds. The rest of the parameters are set to perform well against both adversarial attacker types.



(a) Random fixed attacker, 1 attacker per round (b) Random fixed attacker, 10 attackers per round

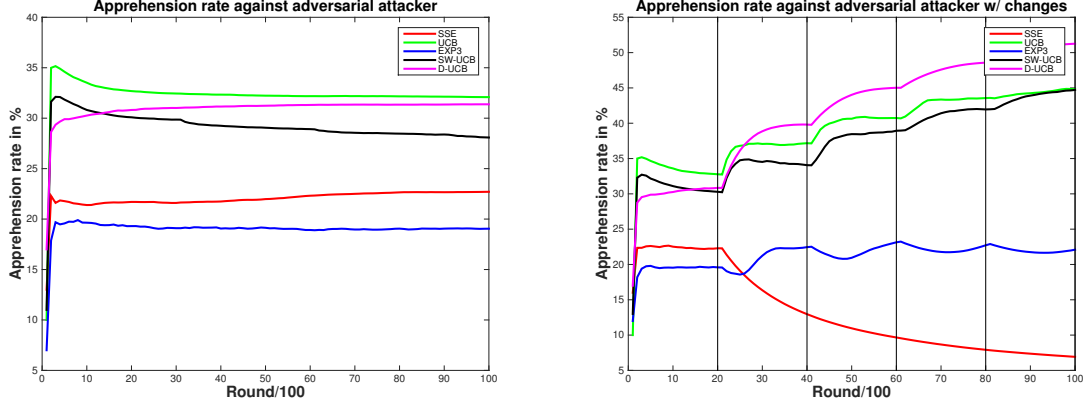


(c) Random fixed attacker with changes, 1 attacker per round (d) Random fixed attacker with changes, 10 attackers per round

Figure 5.1: Defender online learning strategies against random fixed attacker types

In Figures 5.1 there are 4 learning algorithms used for the defender strategy against the random fixed attacker. The widest confidence interval is $\pm 0.72\%$ for Figure 5.1a, $\pm 0.92\%$ for Figure 5.1b, $\pm 0.42\%$ for Figure 5.1c and $\pm 0.56\%$ for Figure 5.1d. We can see that algorithm EXP3 learns faster for the case of 1 attacker per round but in case of 10 attackers per round we see that UCB and SW-UCB learn faster than EXP3. Even though the UCB algorithm is supposed to perform better in stationary environment than EXP3, we can see that it is not always the case and it depends on the level of information the defender can get in every round as seen in Figures 5.1a and 5.1b. For SW-UCB we can clearly see slight decrease of performance every 3000th round, which is the size of the sliding window. In all the Figures 5.1 the D-UCB does not perform very well which is caused by the parameters tuned for the adversarial case. In Figures 5.1c and 5.1d there are the attackers with changes in its strategy probability vectors every 2000 rounds, which is shown by black vertical lines. We can clearly see how the algorithms adapt to the changes. For the case with changes in the attacker strategy the UCB and the SW-UCB algorithms better react to the changes compared to EXP3 algorithm, which is caused by higher exploration of EXP3 algorithm. EXP3 algorithm is designed against adversarial attacker types, however UCB algorithm is

more suitable against stationary attacker types. One can confirm such behavior from the Figures in 5.1.



(a) Adversarial attacker, 1 attacker

(b) Adversarial attacker with changes, 1 attacker

Figure 5.2: Defender online learning strategies against adversarial attacker

We test our learning algorithms against the adversarial attacker who estimates the defender strategy as described in Section 4.1.2. In Figures 5.2 there is performance of learning algorithms against adversarial attacker and adversarial attacker who changes his zone preference vector every 2000 rounds. The widest mean confidence interval is $\pm 0.77\%$ for Figure 5.2a and $\pm 0.98\%$ for Figure 5.2b. In Figure 5.2a we show the efficiency of UCB family of learning algorithms, which outperform the EXP3 learning algorithm. This is partly caused by the UCB synchronization described in Section 4.2.1 caused by the deterministic nature of both players' strategies. This behavior is not realistic and would be broken by one of the player sides (probably by the attacker because the defender does not mind extremely high apprehensions). In Figure 5.2b we can observe the changes in the attacker strategy by the black horizontal lines every 2000 rounds and its impact on the performance of the learning algorithms. In figure 5.2b the UCB algorithms clearly outperforms EXP3 and we can state that they better adapt to the changes. Furthermore one can observe that the changes in attacker strategy even improve the defender payoff when playing UCB type algorithm. This behavior is caused by the learning process of the attacker, if the attacker changes his strategy, the defender adapts his strategy faster than the attacker can learn defender new adapted strategy. Interaction between two players who learn opposite player strategy can be unintuitive, because learning changes the learned subject and vice versa. This is the case when the attacker changes his strategy and unintuitively the defender strategy performance improves. SSE is better than EXP3 but worse than UCB family algorithms, which is caused by the UCB synchronization. In the figure with changes one can see that SSE strategy worsens a lot after the change occurs. This SSE strategy is computed for the initial attacker zone preference vector that is why its performance decreases a lot after changing the attacker zone preference vector. EXP3 has a good ability to adapt to the changes and remain quite stable.

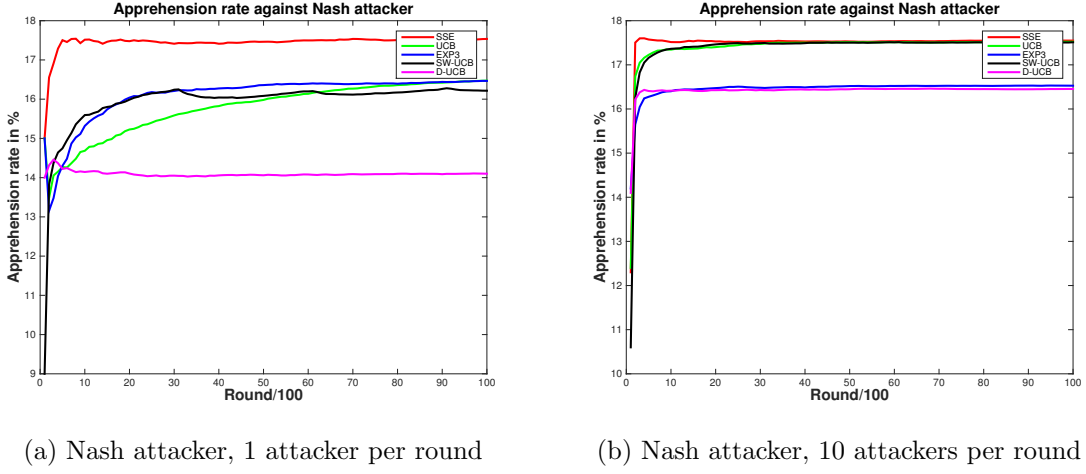


Figure 5.3: Defender online learning strategies against Nash attacker

In Figures 5.3 there are the learning algorithms against attacker who plays Nash equilibrium mixed strategy. The widest mean confidence interval is $\pm 0.58\%$ for Figure 5.3a and $\pm 0.74\%$ for Figure 5.3b. The best performance has intuitively the defender who plays Nash equilibrium as well, which is straightforward. The attacker strategy is not deterministic therefore there is no UCB synchronization. We can also observe that EXP3 has similar performance like UCB family of algorithms. However the algorithm D-UCB algorithm has worse performance, which is caused by discounted observation data and thus the algorithm learns very poorly. In Figure 5.3b we can see that the algorithms learns faster and UCB and SW-UCB gives almost same results as playing the Nash equilibrium in case of 10 attackers per round. Thus we can say that these algorithms converge to the Nash equilibrium in this case. The EXP3 has slightly worse performance due to the 20% exploration ($\gamma = 0.2$). D-UCB algorithm gives similar result to EXP3 algorithm because the parameters are tuned against adversarial attacker and the discount of the past rewards disables the algorithm to learn effectively.

In this section we showed behavior of the proposed learning algorithms and we compared them to each other. We can clearly see the convergence behavior of the learning algorithms. For further experiments we use EXP3 algorithm since it seems more stable than other algorithms and is a good representative algorithm against adversarial strategies. Its stochastic nature is a desirable feature for our model because it makes it more difficult to be exploited or synchronized.

5.2 Experiments with Game-theoretic Strategies

5.2.1 Game-theoretic strategy with error

We do experiments to test an influence of different levels of error ϵ in zone preference vector to compute SSE as described in Section 4.3.2. In Figure 5.4 there are apprehension rates for different levels of error. We observe the performance of SSE strategy for $\epsilon \in [0, 0.2]$. We can see that the adversarial attacker can learn the strategy and through the time the apprehension rate decreases. Especially for higher values of ϵ there is a big decrease in performance. For approximately $\epsilon \geq 0.15$ we get even worse performance than for playing a random defender

strategy, which has the expected payoff 12.5%. In our further experiments we focus on error 0.1. We can see that for SSE without error the performance is still very good even after the attacker learns the strategy. The widest mean confidence interval is $\pm 0.56\%$ for error 0.1.

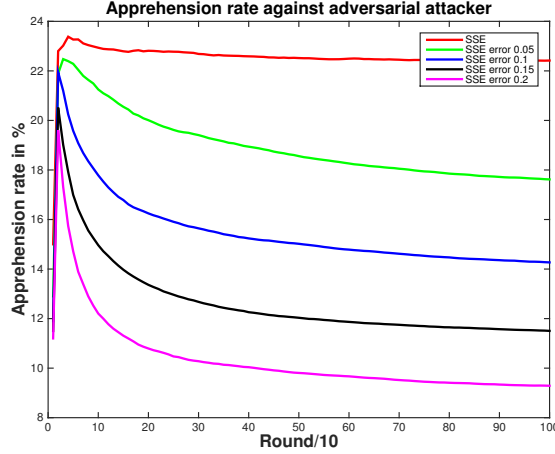


Figure 5.4: SSE strategies with different levels of error against adversarial attacker

5.2.2 Motivating game-theoretic strategy

We perform experiments to test motivating SSE strategy as described in Section 4.3.3. We do experiments for two cases, in the first case we assume that we know the exact Stackelberg equilibrium strategy for the defender and in the second case we do experiments with motivating imprecise Stackelberg equilibrium strategy. We run the experiments 1000 times for each setting. Each game has 1000 rounds.

Precise Stackelberg equilibrium strategy

In this case we assume that the defender knows the exact attacker zone preference vector and thus the defender can compute the exact SSE.

decrease	0	0.005	0.01	0.05	0.075	0.1	0.15	0.2
apprehen.	22.42%	22.91%	23.60%	26.01%	25.06%	23.34%	19.10%	14.15%
confidence	$\pm 0.45\%$	$\pm 0.45\%$	$\pm 0.47\%$	$\pm 0.44\%$	$\pm 0.48\%$	$\pm 0.49\%$	$\pm 0.54\%$	$\pm 0.58\%$

Table 5.11: Motivating SSE with decreased highest value

In Table 5.11 there are apprehension rates for different decrease constants with its confidence intervals. For the decrease constant equals to 0.05 we get the best performance. We can see that it makes sense to do such a modification in SSE strategy because we increase significantly the apprehension rate. In the first column there is unmodified SSE strategy for comparison. On the other hand if we decrease the SSE strategy vector too much we get a poor performance as you can see at the right side of the table.

Imprecise Stackelberg equilibrium strategy

We use the same method of decreasing the highest value in imprecise SSE strategy. We now assume 0.1 error in estimate of the attacker zone preference vector from which the defender computes the SSE strategy.

decrease	0	0.005	0.01	0.05	0.075	0.1	0.15	0.2
apprehen.	14.28%	14.22%	14.51%	15.09%	16.03%	16.40%	16.94%	15.21%
confidence	$\pm 0.57\%$	$\pm 0.59\%$	$\pm 0.58\%$	$\pm 0.60\%$	$\pm 0.61\%$	$\pm 0.61\%$	$\pm 0.57\%$	$\pm 0.54\%$

Table 5.12: Motivating SSE with error with decreased highest value

In Table 5.12 we can see apprehension rates for different levels of decrease constant. If we compare this table with Table 5.11 we can see that we need to motivate the attacker more in case of imprecise equilibrium strategy. The best results are for decrease constant from range (0.075, 0.15). In the first column there is unmodified SSE strategy with error. We can see that the motivating SSE has much higher apprehension rate than unmodified strategy. It is intuitive that we need to motivate the attacker more if we have imprecise SSE strategy.

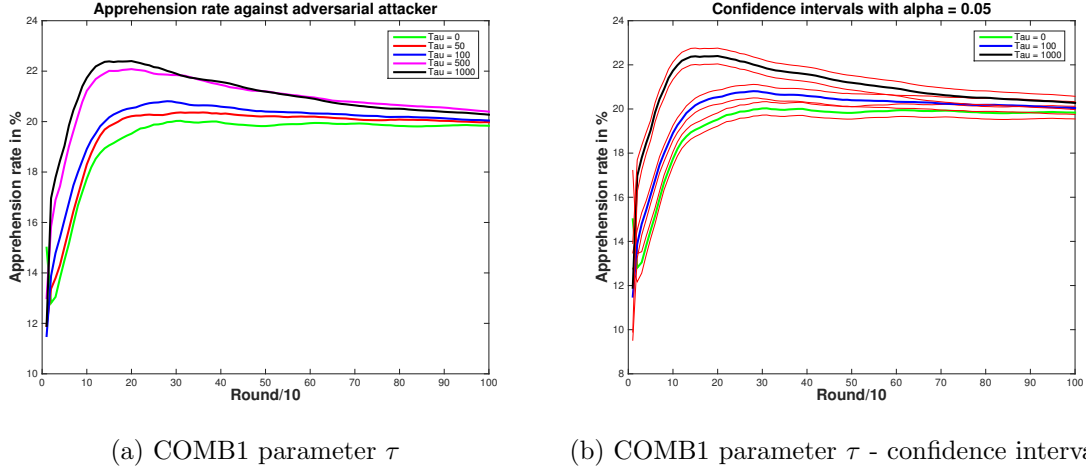
We use this method in our further experiments with combined algorithms in Section 5.4.

5.3 Performance of Combined Algorithms

We described the combined algorithms in Section 4.4. They are based on EXP3 learning algorithm with use of game-theoretic solution. We investigate both cases, a precise game-theoretic solution and an imprecise one. It is known as EXP3 with warm start, which describes the idea that we start the learning algorithm with some prior information, for example the game-theoretic solution. At first we tune the parameter τ of algorithm COMB1, which sets a weight of EXP3 initialization by game-theoretic solution. We present experiments to compare the proposed COMB algorithms with Stackelberg equilibrium strategy and standard EXP3 algorithm against the adversarial attacker. We provide 95% confidence interval.

5.3.1 Parameter tuning

The tuning experiments are done with Stackelberg equilibrium strategy computed from attacker payoff with level of error 0.1 in the zone preference vector. We tune parameter τ of COMB1 algorithm described in Section 4.4, which represents a number of rounds played virtually before the actual start of the game using SSE. We run every experiment 1000 times with 1000 rounds per game. This section also helps to explain the idea of COMB1 algorithm.

Figure 5.5: COMB1 algorithm against adversarial attacker - τ tuning

In Figure 5.5a there are apprehension rate curves for different levels of EXP3 initialization by game-theoretic solution. We can see that for higher values of τ we get higher apprehension rates. In Figure 5.5b there are 95% confidence intervals for three selected values of τ . We can see that stronger initialization improves the defender payoff.

τ	0	50	100	300	500	1000	2000
apprehen.	19.83%	20.02%	20.14%	20.37%	20.42%	20.35%	20.09%
confidence	$\pm 0.28\%$	$\pm 0.28\%$	$\pm 0.28\%$	$\pm 0.29\%$	$\pm 0.29\%$	$\pm 0.30\%$	$\pm 0.31\%$

Table 5.13: Algorithm COMB1 τ parameter tuning against adversarial attacker

τ	0	50	100	300	500	1000	2000
apprehen.	18.23%	18.28%	18.48%	18.16%	17.51%	15.97%	14.35%
confidence	$\pm 0.21\%$	$\pm 0.21\%$	$\pm 0.22\%$	$\pm 0.23\%$	$\pm 0.23\%$	$\pm 0.26\%$	$\pm 0.28\%$

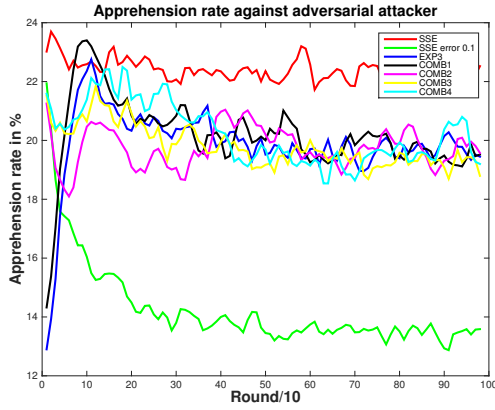
Table 5.14: Algorithm COMB1 τ parameter tuning against adversarial attacker with changes

In Tables 5.13 and 5.14 there is the parameter tuning against adversarial attacker and adversarial attacker with changes. One can observe that for higher τ values we are better off against the adversarial attacker. The error in the estimated Nash equilibrium is quite low so we can expect that stronger initialization brings better performance, but we can also expect that for too high values of τ we restrict EXP3 algorithm and all its properties, which can be a problem against varying opponents. As we can see in Table 5.14 against the adversarial attacker with changes, where we are better off with lower value of parameter τ , because we need the ability of EXP3 algorithm to adapt to abrupt changes in the opponent strategy. For further experiments we use COMB1 algorithm with $\tau = 100$ because we want COMB1 to be robust algorithm against non-stationary attackers. The disadvantage of strong EXP3 initialization (high τ values) is in disabling the desired properties of learning algorithm such as ability to adjust to (learn) new opponent strategies. This experiment supports the idea behind COMB1 algorithm development. We want EXP3 algorithm to converge to optimal

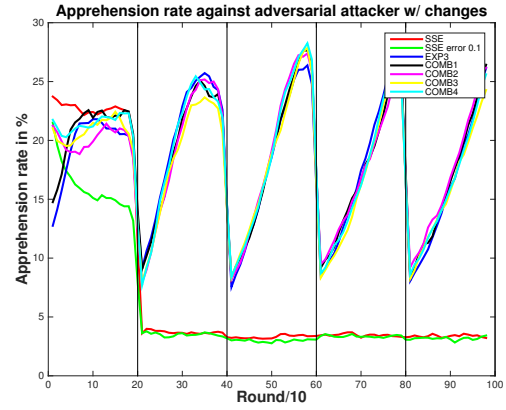
strategy so it makes sense to initialize EXP3 algorithm with Nash equilibrium strategy or with an estimate of it. In the Figures 5.5 one can also see the dynamics of learning. The defender learns the attacker strategy at first so the performance increases and then the attacker also learns the defender strategy so the performance gets more stable. We need to keep in mind the visualization technique of cumulative means. The actual cumulative performance in each round is the derivative in the particular time step of the presented figure.

5.3.2 COMB algorithms performance

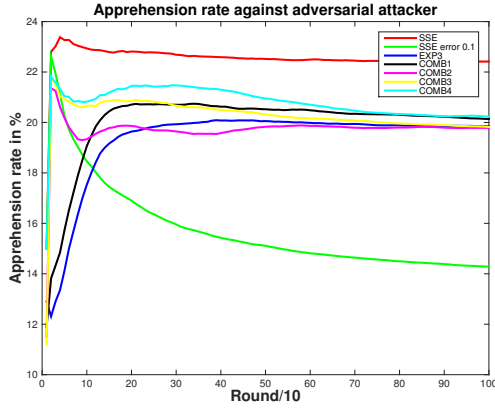
In this section we focus on experiments with the proposed COMB algorithms. We show its effectiveness compared to other approaches. We run each of the experiments 1000 times and each game has 1000 rounds if not stated differently. We provide experiments with game-theoretic solution with error and without error against the adversarial attacker types. As stated before there is a different zone preference vector for the attacker in each experiment of each setting, thus the attacker has different payoff matrix in each experiment. However the defender has always payoff 0 or 1. For comparison we show in the figures EXP3 learning algorithm, Stackelberg equilibrium strategy (SSE) and Stackelberg equilibrium strategy with an error, which is used in COMB algorithms. For each graph we compute a 95% confidence interval and provide a mean interval width across all rounds.



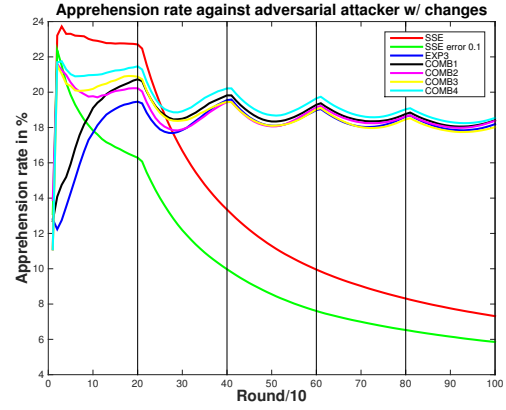
(a) moving averages visualization



(b) moving averages visualization



(c) against adversarial attacker



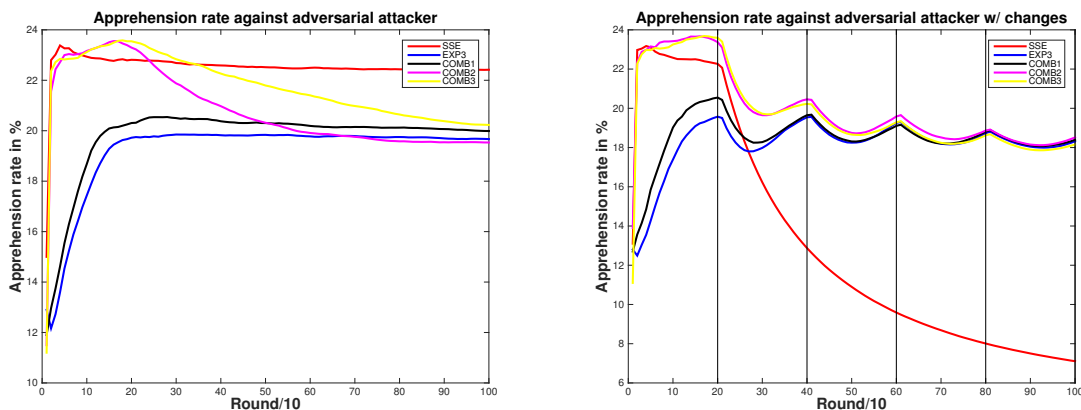
(d) against adversarial attacker w/ changes

Figure 5.6: COMB algorithms with 0.1 error against adversarial attacker

In Figures 5.6 we used another style of result visualization to better understand the behavior of the algorithms. In this thesis we mainly use cumulative averages as described before but it is also interesting to observe visualization by moving averages as shown in Figures 5.6a and 5.6b. We use moving-averages of length 20 rounds. Figure 5.6a shows the same experiment as Figure 5.6c and Figure 5.6b shows the same experiment as Figure 5.6d. From these graphs of same experiments we can observe corresponding behavior. In Figure 5.6b one can observe that SSE with error strategy and SSE strategy have almost same performance after the first change in the attacker zone preference vector, which is obvious because the equilibria are computed for the initial zone preference vector and after the change they perform very poorly. In Figure 5.6c there is a performance of the proposed algorithms against the adversarial attacker. COMB algorithms use game-theoretic solution with an error 0.1. For COMB algorithms the widest confidence interval is $\pm 0.39\%$ and for EXP3 algorithm the width of interval is $\pm 0.30\%$. SSE with error decreases with the attacker learning the strategy. SSE without error obviously gives a very good performance. One can observe that COMB1 has better but very similar performance to EXP3, this comes from the nature of COMB1 algorithm, which is basically the initialized EXP3. Thus we can see that the initialization improves the EXP3. COMB2 algorithm starts with playing SSE with error plus some extra

exploration and then switches permanently to EXP3, we can see that this switch occurs close to the intersection of SSE with error and EXP3 algorithm which is a desired feature of COMB2 algorithm. COMB3 algorithm can arbitrary switch between SSE with error (plus some extra exploration) and EXP3, so we can see that it outperforms the COMB2 algorithm. This is caused by better adaptability to the intelligent attacker. COMB4 has the best performance out of all COMB algorithms and also outperforms EXP3 algorithm. COMB4 similarly to COMB3 can switch between EXP3 algorithm and SSE with error arbitrary but it can choose from 3 SSE with error strategies and thus has better performance. COMB2, COMB3 and especially COMB4 algorithms have very good performance at the first half of the game (up to round 500) and outperform EXP3 and SSE with error. At the end of our game COMB algorithms and EXP3 algorithm have similar performance, which is caused by the attacker learning the defender strategy, also the COMB algorithms tend to play EXP3 later in the game.

We test the COMB algorithms against the adversarial attacker with changes in Figure 5.6d. For COMB algorithms the maximal width of confidence interval is $\pm 0.32\%$ and for EXP3 algorithm the width of interval is $\pm 0.26\%$. This figure shows clearly the advantage of learning algorithm. If we assume that we are not able to detect a change in the attacker payoff and therefore to compute the appropriate game-theoretic solution, we can intuitively expect a poor performance by playing this game-theoretic strategy. Even though these abrupt changes in the attacker payoff matrix are not realistic, we can test algorithms' sensitivity to any changes in the attacker behavior. In this figure there are highlighted the changes in the attacker strategy every 200 rounds by black horizontal lines. We can see that COMB algorithms can successfully adapt to these changes and have very similar total performance as EXP3 algorithm. At the beginning of our game all COMB algorithms are better than EXP3 algorithm. Up to round 200 it is the same figure as Figure 5.6c and then the changes occur. COMB algorithms can adapt to these changes because they make use of EXP3 algorithm and can switch to it in case they need to. So the COMB algorithms remain the desired property of learning algorithms.



(a) against adversarial attacker

(b) against adversarial attacker w/ changes

Figure 5.7: COMB algorithms with no error against adversarial attacker

In Figure 5.7a there are experiments with COMB algorithms using precise game-theoretic solution against the adversarial attacker. The widest confidence interval for COMB algorithm

is $\pm 0.39\%$ and for EXP3 the width is $\pm 0.29\%$. In this figure we do not visualize COMB4 since it boils down to COMB3 in case of precise game-theoretic solution. COMB2 and COMB3 algorithms get even better than SSE strategy, because for the attacker it is more difficult to learn the defender strategy if it is not static. This is partly caused by the extra exploration in COMB algorithm playing SSE, which can confuse the attacker. The attacker learns quite fast a static defender strategy vector SSE. One can observe that even though the COMB2 and COMB3 outperforms the SSE strategy for a short period of time, it then drops in its performance a lot by the attacker eventually learning the strategy. COMBs algorithms decrease under SSE strategy even though they use this SSE strategy, because there is extra exploration 10% added to SSE strategy. This extra exploration causes worse performance than SSE strategy when the attacker learns the defender strategy sufficiently enough. Nevertheless we can see that COMB algorithms outperform EXP3 algorithm at the first half of the game and then they all converge to a similar performance.

In Figure 5.7b we test COMB algorithms using precise game-theoretic solution against the adversarial attacker with changes. For COMB algorithms the widest interval is $\pm 0.32\%$ and for EXP3 algorithm the width of interval is $\pm 0.26\%$. We can see similar behavior as in previous figures. The COMB algorithms can react well to changes in the attacker strategy because of its learning algorithm part. SSE strategy decreases a lot after a change in the attacker zone preference vector as described before. If the defender has a precise SSE strategy he might prefer playing it instead of any other strategy in case of the adversarial attacker however if there are some changes in the attacker payoff matrix the defender would be better off by playing some more sophisticated algorithm like EXP3 or preferably one of the proposed COMB algorithms, because they can adapt to these changes.

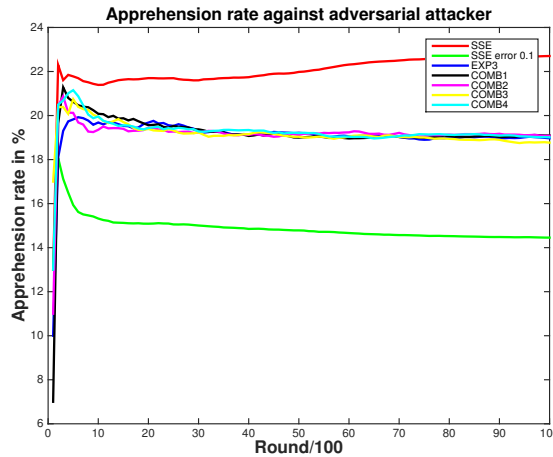


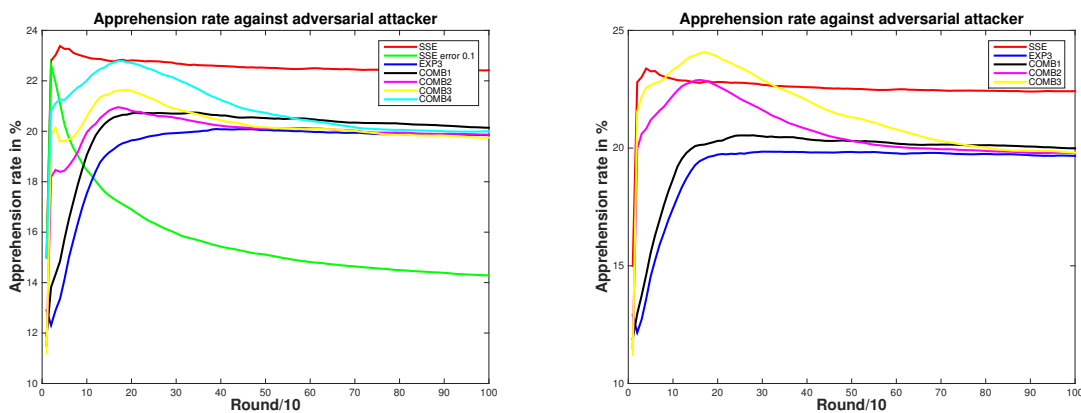
Figure 5.8: Convergence of COMB algorithms with 0.1 error against adversarial attacker

We observe convergence of the proposed algorithm in a longer time window, in Figure 5.8 there are COMB algorithms using game-theoretic solution with error against adversarial attacker for 10000 rounds. This experiment is done 100 times for each setting. The maximal mean width of confidence intervals for COMB algorithms is 0.99% and the width of confidence interval for EXP3 is 0.92%. We can see that COMB algorithms and EXP3 algorithm converge to the same performance quite quickly. Playing precise Stackelberg equilibrium strategy has the best performance however the SSE strategy with 0.1 error gives quite poor result. One

can observe that the precise SSE strategy performance increases during the time, which is caused by the attacker learning more precisely the defender strategy and thus there are more ties in the attacker strategy which the attacker breaks in favor to the defender.

5.4 Performance of Combined Algorithms with Motivating SSE

We analyze combined algorithms, which use motivating SSE strategy. We described the motivating SSE strategy in Section 4.3.3. We do experiments firstly with error 0.1 and secondly without error in defender estimation of the attacker zone preference vector. SSE strategies in the figures do not use the motivating concept and are in the graphs for comparison.



(a) against adversarial attacker, error 0.1, COMBs (b) against adversarial attacker, no error, COMBs using motivating SSE 0.15

Figure 5.9: COMB algorithms against adversarial attacker using motivating SSE

In Figures 5.9 there are experiments with COMB algorithms against the adversarial attacker. There are SSE strategy without error and SSE strategy with error 0.1. The widest mean confidence interval is $\pm 0.37\%$ for COMB algorithms and $\pm 0.3\%$ for EXP3 algorithm in Figure 5.9a. And in the Figure 5.9b the confidence interval is $\pm 0.38\%$ for COMB algorithms and $\pm 0.29\%$ for EXP3 algorithm. In Figure 5.9a there are COMB algorithms using SSE with error 0.1. We can compare this figure with Figure 5.6c where the defender uses COMB algorithms with normal SSE (not motivating). We can see that COMB algorithms with motivating SSE have better performance. At the first half of the game all COMB algorithms outperform EXP3 algorithm and SSE with error strategy. COMB4 approaches to SSE strategy around the round 200. The performance of COMB4 grows up to round 200 because for the attacker it is more difficult to learn changing strategy than in the case of the defender playing fixed strategy vector of SSE. All the algorithms converge to similar performance because the attacker learns the defender strategy and COMB algorithms switch to EXP3 algorithm. SSE strategy is the optimal one and thus it has the best performance. In Figure 5.9b we have COMB algorithms with motivating SSE without any error. If we compare it with the previous figure we can clearly see the improvement in performance due to absence of an error. COMB2 and COMB3 outperforms the SSE strategy for a short period

around round 200. This is caused by higher difficulty for the attacker to learn the defender varying strategy of COMB2 and COMB3 algorithms. COMB1 algorithm is better than EXP3 algorithm due to the initialization by Nash equilibrium. The attacker learns quite fast the defender SSE strategy, which is a fixed mixed strategy vector. From these two graphs we can see that COMB algorithms outperforms the EXP3 algorithm. We can also state that using motivating SSE in COMB algorithms makes sense and gives us better performance.

5.5 Combinatorial Combined Algorithms

In this section we focus on the combinatorial case where the defender assigns multiple resources to the zones in each round. We test combinatorial variants of COMB algorithms which use combinatorial EXP3 as described in Section 3.4.4. We test the algorithms with different number of resources (defenders). We also scale up our model to investigate its behavior for different number of zones. Combinatorial COMB algorithms are very similar to the standard COMB algorithms as described in Section 4.5.

5.5.1 Varying number of defenders

The experiments are done for a game model with 20 zones. We compare the strategies in models with 2, 4, 6 and 8 defenders. These experiments are run 1000 times for each setting and each game has 1000 rounds.

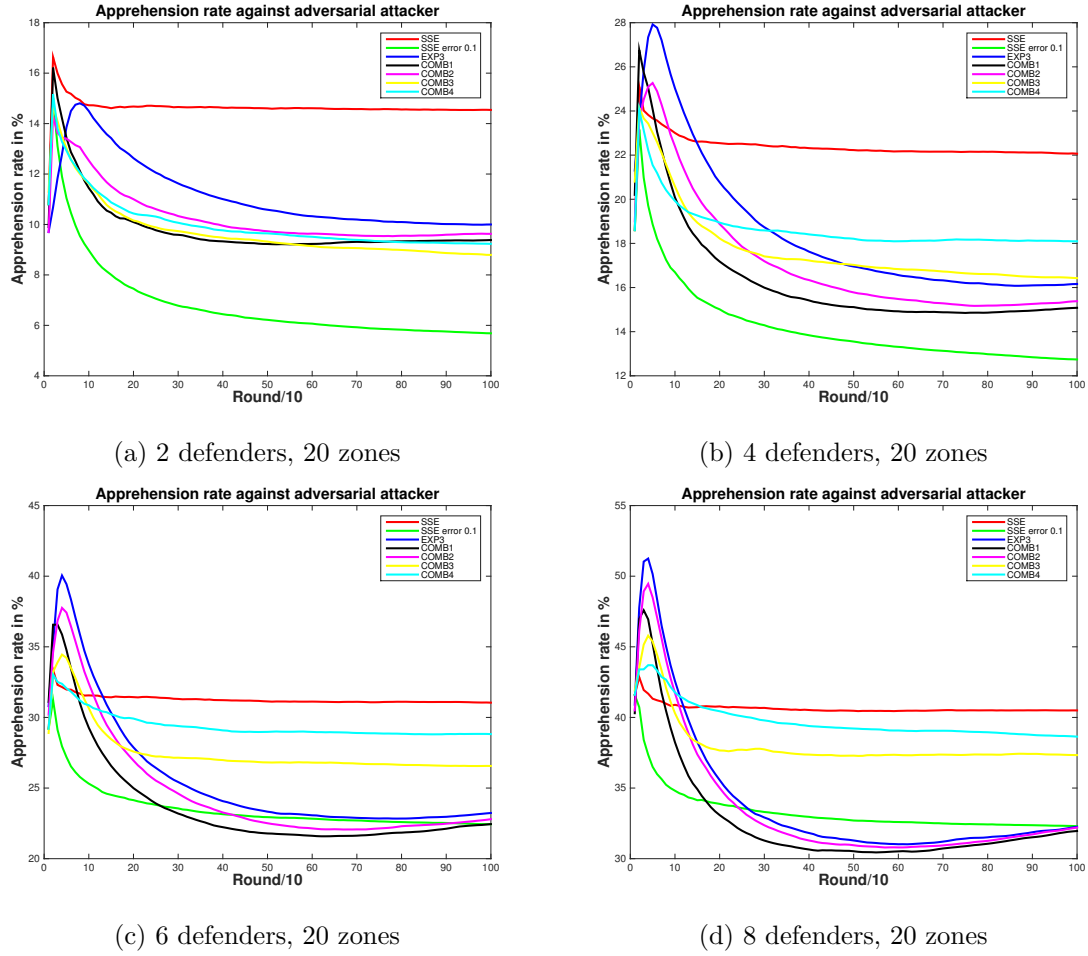


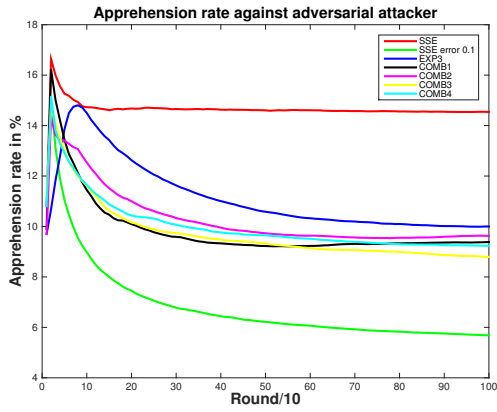
Figure 5.10: COMB algorithms against adversarial attacker, 0.1 error in SSE, varying number of defenders

In Figures 5.10 there are 4 COMB algorithms, SSE with error and SSE without error strategies. In these experiments EXP3 is always the combinatorial version of EXP3. We focus only on the adversarial attacker. The widest mean confidence interval in all the figures is $\pm 0.36\%$ for COMB algorithms, $\pm 0.35\%$ for EXP3 algorithm, $\pm 0.34\%$ for SSE with error strategy and $\pm 0.27\%$ for SSE strategy. One can observe in Figure 5.10a that EXP3 outperforms the COMB algorithms, which is caused by poor performance of SSE with error strategy. EXP3 algorithm gives almost 2 times better performance than SSE with error strategy, because there are too few defenders for too many zones and even a small error in SSE strategy causes low apprehension rate. Due to this fact, the COMB algorithms have worse performance than EXP3. When we increase the number of defenders to 4 in Figure 5.10b, SSE with error does better and so do COMB algorithms. COMB3 outperforms EXP3 algorithm after the half of the game and COMB4 does even better than COMB3, which comes from the nature of the algorithms. COMB4 algorithm chooses from 3 SSE with error strategies and EXP3 while COMB3 algorithm chooses only from 1 SSE with error and EXP3 algorithm. One can observe interesting peaks of the algorithms curves at the beginning of the game, which are caused by increasing number of defenders. The attacker needs time to learn effectively

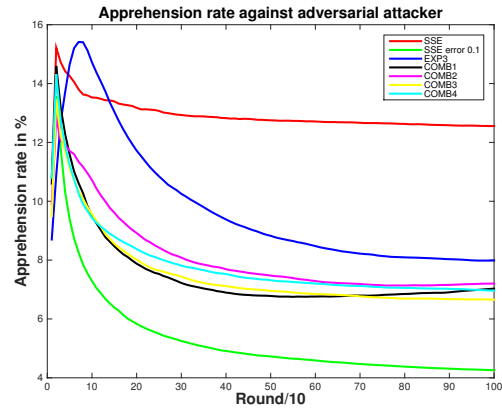
against multiple defenders and at the beginning he plays poorly. However by the steepness of the algorithms curves we can state that the attacker learns very quickly after playing very badly at the beginning. These described features are even stronger with increasing number of defenders in Figure 5.10c and in Figure 5.10d. SSE strategy with error approaches even more to the performance of EXP3 algorithm, because the more defenders there are, the less the error in the SSE strategy vector matters. The defender still chooses the zones with high probabilities even though there are some errors, because these 0.1 errors cannot decrease the real values too much to not be chosen. For the last figure with 8 defenders the SSE with error strategy even outperforms EXP3 algorithm. Nevertheless COMB3 and especially COMB4 algorithms have very strong performance and approach to SSE strategy performance. COMB1 and COMB2 have obvious drawbacks in the limited use of SSE with error strategy. COMB1 use the game-theoretic strategy only to initialize EXP3 and then cannot make use of it anymore and similarly for COMB2 algorithm, which use the game-theoretic strategy at the beginning first couple rounds and then permanently switches to EXP3 algorithm.

5.5.2 Scaling-up experiments

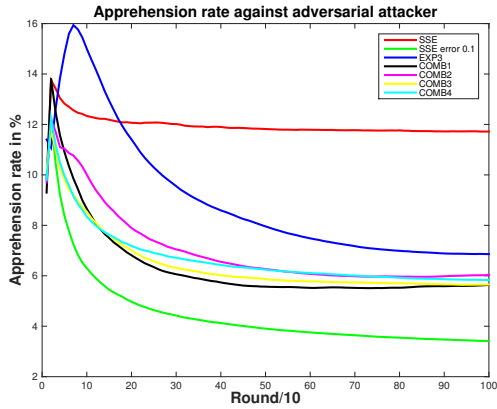
We analyze our model for larger environments. We scale up our model to investigate its behavior for varying number of zones and defenders.



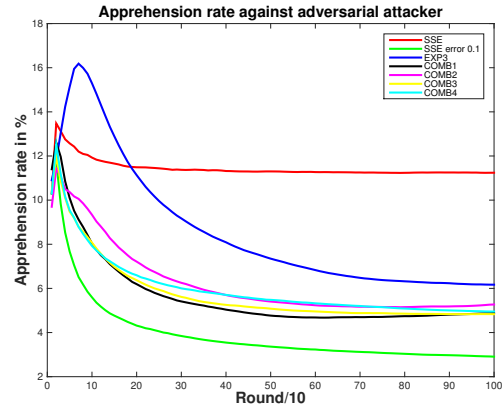
(a) 2 defenders, 20 zones



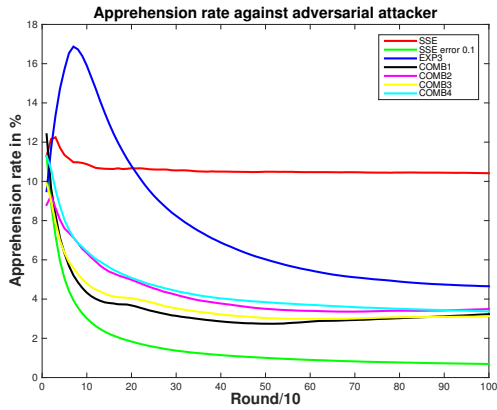
(b) 3 defenders, 30 zones



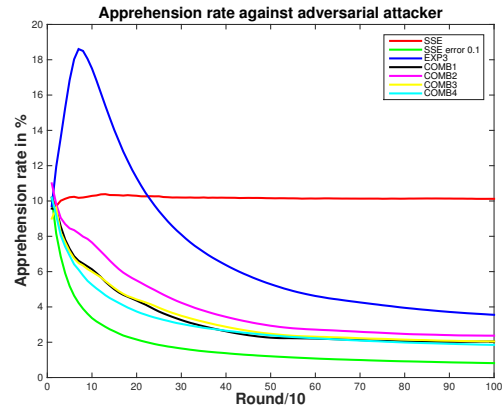
(c) 4 defenders, 40 zones



(d) 5 defenders, 50 zones



(e) 10 defenders, 100 zones



(f) 50 defenders, 500 zones

Figure 5.11: COMB algorithms against adversarial attacker, 0.1 error in SSE, scaling up

In Figures 5.11 there are experiments for 2 defenders and 20 zones, 3 defenders and 30 zones, 4 defenders and 40 zones, 5 defenders and 50 zones, 10 defenders and 100 zones and 50 defenders and 500 zones. The widest mean confidence interval in all the figures is $\pm 0.21\%$ for COMB algorithms, $\pm 0.18\%$ for EXP3 algorithm, $\pm 0.22\%$ for SSE with error strategy

and $\pm 0.19\%$ for SSE strategy. One can observe very similar behavior of each strategy in all the figures. However there are differences worth mentioning. We can see that for increasing number of zones the performance of all algorithms decrease and even the performance of SSE strategy worsens, which is caused by higher absolute number of zones even though we cover them with relatively same number of defenders. If there are more zones to choose from, it is more difficult to apprehend the attacker, because the attacker can choose from more zones which can be same favorable to him. There is always an error in the attacker estimate of defender strategy that is why the attacker is more sensitive to choose a zone which he would not choose in precise (full knowledge) Stackelberg game and which gives him lower payoff. Therefore the attacker visits zones, which the defender does not surveil because he assumes they are not likely to be chosen by the attacker. Interesting fact to mention is that the differences between SSE strategy and SSE with error strategy remains almost the same and also the differences between SSE strategy and EXP3 algorithm are almost the same in all these experiments so we can state that the model behaves very similarly even for larger systems. Therefore we can expect that our conclusions about COMB algorithms derived for small models hold for larger games in similar way. We can see that EXP3 algorithm performance increases a lot at the beginning of the game for the increasing number of zones. This is caused by the increasing difficulty for the attacker to learn the defender strategy. For more zones and more defenders the attacker needs to explore more possibilities and thus the learning takes longer time that is why the defender is very successful at the beginning. However as mentioned before once the attacker learns the defender strategy, it is more difficult for the defender to apprehend the attacker for increasing number of zones and that is why the performance of defender strategies decreases with more zones as we can see in the figures.

Chapter 6

Conclusion

In this thesis we reviewed the recent literature on security games and online learning. We discussed scientific papers in the area and described the connections to this thesis. We mentioned and described several online learning algorithms. Then we focused on the game-theoretic approach to the security games. We proposed a sample security game model for border patrol, which we use to analyze various game strategies. There is an empirical performance evaluation of the game-theoretic and online learning algorithms against multiple attacker types. We study usability of these approaches under various conditions.

We presented a modification to the game-theoretic solution where we motivate the attacker to choose a zone, which the defender wants him to choose. This modification called motivating SSE improved the performance of the game-theoretic solution against adversarial attacker. We showed that if the attacker does not know precisely the defender strategy, he might not play optimally. Therefore one should keep in mind such probable behavior of Stackelberg security game model in real-world applications, where there is an uncertainty in the attacker estimate of the defender strategy. Motivating SSE strategy is a straightforward approach to partly handle such uncertainty. Another pitfall one can derive from the proposed game model is the use of two deterministic strategies against each other. We showed a synchronization (exploitation) of two deterministic learning algorithms, UCB algorithm on the defender side and a type of fictitious play on the attacker side. During designing a game model one should be aware of this possible vulnerability using a deterministic strategy. This observation holds for vast majority of security games, where it is not desirable to use deterministic strategies, because they can be exploited by the opponent.

The main novelty of this thesis are the combined algorithms, which connect online learning algorithm EXP3 with game-theoretic solution. We proposed 4 combined algorithms, which differ by the combining method, adaptability to the attacker strategy and level of use of game-theoretic solution. They are called COMB1, COMB2, COMB3 and COMB4. We also use motivating game-theoretic solution in COMB algorithms. We differ our experiments to two main branches; firstly we investigated the standard case, where there is 1 defender resource per round and secondly we focused on the combinatorial case, where we assume that the defender has more resources. These two concepts differ by used online learning algorithm. Based on the type of the learning algorithm we derived COMB algorithms for both concepts. We tested these combined algorithms against various types of game setting. There are experiments with various numbers of defenders and zones. We also showed scalability of our proposed algorithms, where we demonstrated usability of the combination algorithms for larger games.

Combined algorithms present an effective approach to security games and make use of advantages of online learning algorithm and game-theoretic solution. The advantage of learning algorithm is its adaptability to various attacker types and its robustness in varying environments. On the other hand, the game-theoretic solution provides mathematical approach to security games, where it makes use of estimated game properties such as attacker payoff matrix using game theory. We showed that COMB algorithms have a very good performance in the non-combinatorial case and are better than EXP3 online learning algorithm or an imprecise game-theoretic solution. For the combinatorial case, we are better off using EXP3 algorithm if we have lower number of resources (defenders), but with increasing number of resources we are better off using COMB3 or COMB4 algorithm. However even in the cases where EXP3 outperforms COMB algorithms, the COMB algorithms still have a very good performance due to using EXP3 as a part of them. Therefore we can use them without worries of getting much worse performance than online learning algorithm. We also showed that in a long-term game all the COMB algorithms converge to EXP3 algorithm. One can derive it makes sense to use a learning algorithm with some extra information in a form of the game-theoretic strategy. It is called a warm start, when the learning algorithm starts with some prior information about the opponent's strategy. However one need to be careful with the use of the imprecise game-theoretic solution, because obviously if the error in the game-theoretic solution is too high it can harm the performance. Our proposed combined algorithms can prevent the use of very imprecise game-theoretic solution by switching to online learning algorithm.

In this thesis we studied and analyzed a restricted model of border patrol problem. However the model uses some general concepts and ideas, which make the conclusions applicable to many other domains. The model can be deployed in areas where there is a need to use resources effectively with frequent interaction between the players and with uncertainties about opponent strategy. Technically speaking, models where there is a version of multi-armed bandit problem.

There are several restrictions of the model. We assume a fixed defender payoff structure; the defender is indifferent among the zones (actions). In many real-world applications there are preferences for actions, which represent how difficult is to secure a particular zone. In the security domain, there are sometimes targets, which are easily kept under surveillance and some which are not. We also limit COMB algorithms to use only EXP3 online learning algorithm.

6.1 Future Work

There are several possible future directions. First direction could focus on a formal analysis of the proposed combined algorithms, one could derive and prove the regret bounds, which would guarantee the algorithm performance and would make the algorithms formally comparable to the other learning algorithms. One could also investigate convergence of the proposed algorithms to known equilibrium concepts. Another direction could aim for different use of a game-theoretic solution. In [25] the authors propose a robust game-theoretic algorithm, which can handle multiple types of uncertainties. If there would be a model considering different types of uncertainties, this robust algorithm could be used in our combined algorithms as the game-theoretic part, which would better reflect these uncertainties. However one should be aware that this robust algorithm can be only suboptimal. Another direction of possible

future work is bringing defender action preferences into the game model, which would better reflect real-world applications. There might be more attacker types analyzed, for example the attacker could also use another online learning algorithms to investigate performance of the game model against different possible attacker types, which would make the model better applicable to various domains.

Appendix A

Content of DVD

Attached DVD contains source files in Matlab of the proposed game model. All the experiments done in this thesis come from this game model. The basic model on the DVD can be easily set to any setting described in this thesis. We also attach algorithm LTRA written in Java for solving Stackelberg games introduced in [21], which we used for our experiments.

DVD contains this directory tree:

```
master_thesis_Klima
├── thesis
│   ├── thesis_LaTeX_source_code
│   └── thesis_PDF
├── model
│   ├── game_model
│   └── solving_Stackelberg
```

A.1 Game Model

Game model is implemented in Matlab. Early framework of zero-sum game, which modeled the basic interaction between the defender and the attacker came from other students at the University of Texas at El Paso, however all the defender and the attacker strategies (online learning algorithms, game-theoretic strategies, combined algorithms, fictitious play), system of multiple defenders, general-sum game model, etc. were implemented by the author of this thesis and the rest of the code was modified to fit our game model. There are several classes of the program. Basic model can be used only by setting variables in classes ModelMain.m and BorderVars.m. All the other classes are the core classes of the model and does not have to be modified.

- ModelMain.m
Main class of the program. The user operates this class, where he chooses defender and attacker strategy. According to the type of the experiment he chooses generated matrices of zone preference vectors for the attacker and Stackelberg equilibrium vectors for the defender. These matrices are needed only if there is the adversarial attacker and a defender who uses game-theoretic solution.

- `BorderVars.m`
The user can set all the parameters of the model and of all the algorithms used. In this class one can set number of attackers, defenders, zones or parameters of learning algorithms, combined algorithms etc.
- `DefStrat.m`
This class contains all the defender strategies.
- `AttStrat.m`
This class contains all the attacker strategies.
- `BorderModel.m`
This class simulates the whole game.
- `Simulation.m`
This class simulates each round in the game.
- `ZoneLoc.m`
This class allocates a zone to the player randomly according to the player's strategy probability distribution vector. Allocated zone is the chosen zone by the player in every round.
- `ZoneLoc2.m`
This class is a combinatorial version of `ZoneLoc.m` class, so it allocates multiple zones to the player.
- `MISC classes`
This folder contains other classes which are used for visualization or computing confidence intervals.

We add a folder with matrices of zone preference vectors for the attacker and Stackelberg equilibria for the defender for the basic experiment of 8 zones and 1 defender and level of error 0.1 in the zone preference vector. For other experiments it is needed to generate new matrices with appropriate settings by running the program in Java for solving Stackelberg game.

A.2 Algorithm for Solving Stackelberg Games

On the DVD there is algorithm LTRA (Linear-Time Resource Allocation) implemented in Java for solving Stackelberg games. We use this program for computing Stackelberg equilibria, which we use as input to our game model. In class `LtarMain.java` we set the game properties as number of zones, number of targets (zones in our case) and number of resources (defenders). In `StructuredSecurityGame.java` we can set the error in zone preference estimate. After running the `LtarMain.java` class there are two matrices generated. First matrix contains the Stackelberg equilibria for the defender and the second matrix contains the zone preference vectors for the attacker. We use these matrices as input to our game model.

Bibliography

- [1] 2012–2016 border patrol strategic plan. U.S. Customs and Border Protection, 2012.
- [2] B. An, M. Brown, Y. Vorobeychik, and M. Tambe. Security games with surveillance cost and optimal timing of attack execution. *AAMAS*, 2013.
- [3] B. An, C. Kiekintveld, E. Shieh, S. Singh, M. Tambe, and Y. Vorobeychik. Security games with limited surveillance. *AAAI*, 2012.
- [4] P. Auer, N. Cesa-Bianchi, and P. Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 47, 235-256, 2002.
- [5] P. Auer, N. Cesa-Bianchi, Y. Freund, and R. E. Schapire. Gambling in a rigged casino: The adversarial multi-armed bandit problem. *NeuroCOLT2*, 1998.
- [6] P. Auer, N. Cesa-Bianchi, Y. Freund, and R. E. Schapire. The non-stochastic multi-armed bandit problem. *SIAM Journal on Computing*, 32(1), 2001.
- [7] A. Blum, H. Nika, and A. D. Procaccia. Lazy defenders are almost optimal against diligent attackers. *AAAI*, 2014.
- [8] S. Bubeck and N. Cesa-Bianchi. Regret analysis of stochastic and nonstochastic multi-armed bandit problems. *Foundations and Trends in Machine Learning*, 2012.
- [9] N. Cesa-Bianchi and G. Lugosi. Combinatorial bandits. *Journal of Computer and System Sciences*, 2011.
- [10] R. Combes. Email correspondence on projection algorithm for combinatorial EXP3 algorithm, March 2015.
- [11] R. Combes, M. Lelarge, A. Proutiere, and M. S. Talebi. Stochastic and adversarial combinatorial bandits. 2015.
- [12] P. I. Cowling, E. J. Powley, and D. Whitehouse. Information set monte carlo tree search. *IEEE Transaction on Computational Intelligence and AI in Games*, 2012.
- [13] C. Daskalakis, R. Frongillo, and al. On learning algorithms for nash equilibrium. *SAGT*, p. 114 - 125, 2010.
- [14] Y. Freund and R. E. Shapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 1997.

- [15] D. Fudenberg and D. K. Levine. *The Theory of Learning in Games*. The MIT Press, 1998.
- [16] A. Garivier and E. Moulines. On upper-confidence bound policies for non-stationary bandit problems. Technical report, 2008.
- [17] C. Kiekintveld, M. Jain, J. Tsai, J. Pita, F. Ordonez, and M. Tambe. Computing optimal randomized resource allocations for massive security games. In *AAMAS-09*, 2009.
- [18] R. Klima, C. Kiekintveld, and V. Lisy. Online learning methods for border patrol resource allocation. *GAMESEC*, 2014.
- [19] D. Korzhyk, V. Conitzer, and R. Parr. Solving stackelberg games with uncertain observability. *AAMAS*, 2011.
- [20] L. Lai, T. and H. Robbins. Asymptotically efficient adaptive allocation rules. *Advances in Applied Mathematics* 6, (4-22), 1985.
- [21] O. Lerma, V. Kreinovich, and C. Kiekintveld. Linear-time resource allocation in security games with identical fully protective resources. *Association for the Advancement of Artificial Intelligence*, 2011.
- [22] J. Letchford, V. Conitzer, and K. Munagala. Learning and approximating the optimal strategy to commit to. *Bellairs Workshop on Algorithmic Game Theory*, 2009.
- [23] J. Nash. Non-cooperative games. *The Annals of Mathematics*, 54(2):286–295, 1951.
- [24] K. C. Nguyen and T. A. T. Basar. Security games with incomplete information. In *Proc. of IEEE Intl. Conf. on Communications (ICC 2009)*, 2009.
- [25] T. H. Nguyen, A. Jiang, and M. Tambe. Stop the compartmentalization: Unified robust algorithms for handling uncertainties in security games. *Proceedings of the 13th AAMAS*, 2014.
- [26] H. Robbins. Some aspects of the sequential design of experiments. *Bulletin American Mathematical Society* 55:527–535, 1952.
- [27] L. S. Shapley. Some topics in two-person games. *Advances in game theory*, 1964.
- [28] Y. Shoham and K. Leyton-Brown. *Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations*. Cambridge University Press, 2009.
- [29] M. Tambe. *Security and Game Theory: Algorithms, Deployed Systems, Lessons Learned*. Cambridge University Press, 2011.
- [30] J. Tsai, Z. Yin, J.-y. Kwak, D. Kempe, C. Kiekintveld, and M. Tambe. Urban security: Game-theoretic resource allocation in networked physical domains. *Association for the Advancement of Artificial Intelligence*, 2010.
- [31] R. Yang, B. Ford, M. Tambe, and A. Lemieux. Adaptive resource allocation for wildlife protection against illegal poachers. *AAMAS*, 2014.

- [32] Z. Yin, M. Jain, M. Tambe, and F. Ordonez. Risk-averse strategies for security games with execution and observational uncertainty. In *AAAI-11*, 2011.
- [33] P. Young, H. The possible and the impossible in multi-agent learning. *Elsevier B.V.*, 2007.