2013-01-01

Security Games with Interval Uncertainty

Md Towhidul Islam
University of Texas at El Paso, mislam2@miners.utep.edu

SECURITY GAMES WITH INTERVAL UNCERTAINTY

MD TOWHIDUL ISLAM

Department of Computer Science

APPROVED:

Christopher Kiekintveld, Chair, Ph.D.

Vladik Kreinovich, Ph.D.

Eric Smith, Ph.D.

Benjamin C. Flores, Ph.D.
Dean of the Graduate School

to my

MOTHER and FATHER

with love

SECURITY GAMES WITH INTERVAL UNCERTAINTY

by

MD TOWHIDUL ISLAM

THESIS

Presented to the Faculty of the Graduate School of

The University of Texas at El Paso

in Partial Fulfillment

of the Requirements

for the Degree of

MASTER OF SCIENCE

Department of Computer Science

THE UNIVERSITY OF TEXAS AT EL PASO

August 2013

Acknowledgements

I would like to express my deepest appreciation to Dr. Christopher Kiekintveld for all his supports as an advisor and mentor throughout the whole MS program. My gratitude also goes to Dr. Vladik Krenovich for all his advice as a teacher and mentor and all the helpful feedback all these years. Many thanks to Dr. Eric Smith for his helpful comments and being my committee member.

Special thanks go to Oscar Veliz for his huge support as a friend and for giving me the valued feedback on this report. Thanks to my lab members, Curtis Chambers, and Eric Gutierrez, for their helps from time to time.

Finally I would like to thank all my family members for all their sacrifices. My deepest thanks go to my friends, Pallab Karim, Abu Jahid, Sajib Barman, Rashedul Hasan, Sariful Islam, and Md Rajib for their company and supports during this whole process.

Abstract

Game theory has become an important tool in solving real-life decision making problems. Security games use the concept of game theory in adversarial scenarios to protect critical infrastructure. The main purpose of security games is to allocate security resources among various targets and maximize payoff for the defender considering various kinds of attackers. It is hard for domain experts to predict the attacker's behavior, so one of the major challenges in describing this game model is representing uncertainty about the attacker's payoff. Several approaches have been developed to generate these game models based on uncertainty, such as Bayesian games. However Bayesian approaches have drawbacks in solution quality and time. The work of this thesis proposes a polynomial time algorithm that represents uncertainty based on intervals and generates a robust solution for large security games unlike previous methods. I also present a methodology to transform Bayesian games with distributional uncertainty into interval games, and use this novel interval algorithm to generate an approximate solution. At the end of this thesis, empirical data shows that this novel technique is faster and generates higher quality solution compared to previous Bayesian approaches.

Table of Contents

Chapter 1

Introduction

In every nation, protecting critical infrastructure is one of the main tasks of security personnel. However, compared to the number of targets, the number of security resources is typically limited. Security organizations must randomize and allocate their limited resources to minimize the risk of a target being attacked. Security games can play a big role in solving these resource allocation problems. Applications based on security games like ARMOR, IRIS, and PROTECT [25, 28, 29] have already been deployed in the field and are expediting resource allocation decisions.

In the process of developing security games models, one of the most challenging tasks is to define the characteristics of adversaries or attackers. As decision makers are not aware of the behaviors of the potential attackers, a model need to be developed to represent uncertainty. Previous works on security game models based on uncertainty lead to solution methods which lacks guarantees on solution quality or are computationally expensive. In the work of this thesis, a fast polynomial time algorithm is presented for security games which handles uncertainty efficiently by representing the uncertainty as intervals over possible payoffs. Experiments show that this novel interval technique is competitive compared to existent solution methods and for large games it performs better.

1.1 Game Theory

In game theory [20], a game is a mathematical model which consists of a set of players, a set of actions for each player and a set of reward values for each of the outcomes. An outcome is an action choice for each player. Game theory is the study of game models where there

Table 1.1: A zero-sum game

	c	d
a	$(1, -1)$	$(3, -3)$
b	$(0, 0)$	$(2, -2)$

are conflicts or cooperations between intelligent rational decision makers. Game theory is applied to various fields, explicitly in economics, political science, psychology, biology, computer science and logic. In game theoretic analysis of a given problem, a game model is generated for that problem based on data or expert knowledge of the domain, and then the game is analyzed to determine good action choices or to predict what other players are likely to do.

Some popular examples of games are The Prisoner's Dilemma, Battle of Sexes, and Hawk-Dove games. Table 1.1 contains an example of a zero-sum game where row player chooses from actions a and b, and the column player chooses from c and d. There are four outcomes in this game. Each of the outcomes defines reward values for each player for playing actions. For example, the outcome (a, c) defines that if row player plays action a and column player plays action c, reward values for them are 1 and -1, respectively. The main characteristic of a zero-sum game is that for each possible outcome the reward values for two players sum up to zero.

Strategies of a player can be divided into two categories: pure strategies and mixed strategies. In the pure strategy game, the player selects one action from the set of actions. In the mixed strategy game, the player selects an action based on some probability distribution. Playing a mixed strategy has advantages over a pure strategy, which is explained in more detail in Chapter 3.

Stackelberg games are special type of games which consist of two players, a leader and a follower. The leader acts first and the follower acts next based on the observation of the leader's strategy. The Stackelberg game model is being used as a framework to develop

Table 1.2: A non-cooperative non-zero-sum game

	c	d
a	$(2, 1)$	$(4, 0)$
b	$(1, 0)$	$(3, 1)$

many game-theoretic solutions specially in security domains, because it is a natural model of attacker surveillance and planning capabilities.

Some of the common solution concepts associated with game theory are best response, Nash Equilibrium, and Strong Stackelberg Equilibrium. The following section describes briefly these solution concepts which will be used later in this thesis.

1.1.1 Best Response

Best response is a strategy which produces best outcome for a player given the strategy of the opposite player. In the example of Table 1.2, if the row player plays choice a, the *best response* for the column player would be c, as changing the decision does not improve the column player's reward value. However, if the row player opts for b, the best response for the column player would be d for similar reason.

Now suppose that the row player opts for a mixed strategy, and instead of choosing a or b, he chooses each of them 50% of the time. If the attacker plays c observing the defender's actions, he receives a payoff of 0.5 $(1 \cdot 0.5 + 0 \cdot 0.5)$ and by playing d he receives 1 $(0 \cdot 0.5 + 2 \cdot 0.5)$. In this scenario, the attacker's best repose will be playing choice d.

1.1.2 Nash Equilibrium

The concept of Nash equilibrium was first proposed by the famous mathematician John Forbes Nash Jr. [20]. It is a solution concept where players move simultaneously and each of the opposite players plays their best responses knowing the actions of other player

and they do not benefit by changing their own actions. Table 1.2 shows an example of a non-cooperative game where the row and the column players play against each other.

In this game, if the row player plays a and the column player plays c, none of them can benefit by changing their respective choices. The only Nash-Equilibrium outcome in this game is when row player plays a and column player plays c. On the other hand, the outcome (b, d), where the row player and the column player plays b and d respectively, is not considered as an Nash-Equilibrium outcome as the row player has a reward of 3, but can benefit by playing a, where he would have a reward value of 4.

1.1.3 Strong Stackelberg Equilibrium

Strong Stackelberg equilibrium (SSE) is a solution concept for Stackelberg games where the leader plays a mixed strategy and the follower observes the mixed strategy and chooses a best response. For example suppose that in the game of Table 1.2, the row player plays a 50% of time and b 50% of time. The best response for the column player is d, which gives the row player an expected utility of 3.5 and the column player gets 0.5. To find the SSE, all the possible mixed strategies must be searched for the leader to find the one that gives the maximum value. In this solution concept, it is assumed that the follower breaks tie in favor of the leader, which means if the follower's chosen strategy returns him reward values similar to the leader's, follower will play a different choice where the leader gets better reward value.

1.2 Security Games

Security games are a branch of game theory where adversarial entities try to maximize their own payoffs or reward values by playing against the other player. In particular, security games model problems in allocating security resources efficiently against motivated attackers who can learn security strategies. In this field, one of the biggest challenges is creating game models based on unreliable data or conflicting expert opinions. As most of

Table 1.3: A security game with 4 targets

	t_1	t_2	t_3	t_4
Defender's Reward for Covered Target	1	0	2	3
Defender's Penalty for Uncovered Target	-1	-4	-6	10
Attackers's Penalty for Covered Target	-2	-3	-3	-2
Attackers's Reward for Uncovered Target	1	3	4	5

these game models have some associated uncertainty, it is hard to find a robust solution for these games.

Table 1.3 contains a simple example of a security game where each column represents possible targets, where a target can be a critical infrastructure (e.g., terminals in airport). In any security game scenario, there are limited number of defenders (usually less than the number of targets). In the example of Table 1.3 there is only one defender. When the defender protects target t_1, the target t_1 is covered. For each target there is a reward and penalty value for that target for both players for the cases when the target is either covered or uncovered at the time of attack.

As this game is modeled as a Stackelberg game, the attacker can observe the randomized strategies of the defender. When t_1 is covered, if the attacker chooses to attack t_1, the attacker gets a penalty or lower payoff, but if target t_1 is uncovered, he receives a higher reward value. The task for the defender is to find the optimal randomized strategy, assuming that the attacker will respond optimally. This corresponds to calculating the Strong Stackelberg Equilibrium (SBE) of the game.

Security games are being used as a general framework for modeling a wide variety of resources allocation decisions in adversarial security domains. In decision making, Stackelberg games have become an important modeling paradigm which is more analogous to a real-life security scenarios where attackers can observe the strategies of the defender and can act accordingly. The technique of Stackelberg games are also being used in security

domains such as critical infrastructure protection [5, 27], computer networks [3, 22], and robot patrolling strategies [2, 4, 9].

1.3 Security Games with Uncertainty

Due to unpredictability of human attackers security games must be modeled based on uncertainty. Bayesian games are [11] the most common framework for reasoning about uncertainty in game theory. In this formalism, information about the characteristics of one player or more is incomplete. A Bayesian game models uncertainty by introducing "Nature" as a player in a game. Nature introduces uncertainty by taking random moves that can not be observed fully by the other players, analogous to a dealer in Poker choosing the cards in each player's hand randomly. The existing approaches to solve security games with uncertainty are based on Bayesian Stackleberg game formulations that model uncertainty about payoffs, the observation capabilities of an attacker and other factors [12, 16, 23, 24, 31]. All of these current solution methods have problems with solution quality and solution time. Finite Bayesian Stackelberg games where there are finite number of attacker types are NP-hard to solve [8], and infinite Bayesian Stackelberg game does not have an exact algorithm [16], and lacks guarantees in solution quality.

The existing model of Infinite Bayesian Stackelberg games uses distributions (e.g., Gaussian distribution or uniform distribution) to define uncertainty. This thesis uses intervals to represent possible payoff uncertainty. It takes a worst-case optimization approach which tries to maximize the defender's worst case payoff with respect to uncertainty. In this model, the defender only knows that the attacker's payoff for a target falls within some interval of possible values and tries to maximize the worst case payoff with respect to these intervals.

Interval-based approach has advantages over the Bayesian approach for modeling uncertainty in both solution quality and time. It is much easier for a domain expert to generate an interval game model compared to Bayesian model, because this does not require de-

tailed informations about the probability distribution of payoffs for possible targets. While existing Bayesian models are NP-hard, a polynomial-time approximation algorithm is introduced that is scalable and provides tight bounds on solution quality for the model with interval uncertainty. For large security games, this algorithm based on interval uncertainty may be the only computationally feasible approach to handle uncertainty.

1.4 Contributions

My primary contributions in the work of this thesis are as follows:

1. Definition of the new model of Interval Security Game (ISG).
2. Analysis of the model, leading to the implementation of a polynomial-time algorithm for finding solutions to ISG.
3. A methodology for approximating Distributional Security Games (DSG) by transforming distributional uncertainty to intervals.
4. Experiments to evaluate the efficiency of the new algorithm and the solution quality, including comparisons with the best known methods for DSG showing that my transformation methods yields high quality approximate solutions.
5. A new methodology for defining and experimenting on incorrectly specified game models, and experiments further showing the robustness of the interval model in cases of incorrect modeling assumptions by the defender.

1.5 Thesis Outline

Chapter 2 describes related works on the basic ideas based on which this thesis was formed. These include security games, modeling uncertainty, worst case approaches, works on interval and applications of security games deployed in various fields. Chapter 3 starts

with a formal definition of security games and extends this to Interval Security Games. Chapter 4 presented the ISG algorithm and all the ideas and formulations for this new approach. Chapter 5 describes the main concept used in distributional security games and then describes how these games can be transformed into ISG. Chapter 6 presents all the experimental analysis conducted to evaluate the solution quality and runtime of the new model and algorithm. Chapter 7 concludes the thesis with a summary of the work done and topics for possible future work.

Chapter 2

Related Work

In this chapter I briefly describe some game solving approaches in security games, their backgrounds and applications in the real world.

2.1 Game Theory Applied to Security

Nations and organizations need to protect locations of military, economic and political importance from groups or individuals that can cause harm. Due to the fact that there are limited number of security resources, full coverage of important locations is not possible. Security game are being used to create optimal patrolling strategies to protect targets from attackers. There are several applications deployed at various locations for security that are described in this section.

The Assistant for Randomized Monitoring over Routes (ARMOR) system is deployed at Los Angeles International Airport (LAX) since 2007 to schedule canine patrol and vehicle checkpoints at airport [25]. Figure 2.1(a) is one of the LAX checkpoints scheduled by the ARMOR system. ARMOR models patrolling or monitoring problem as a Bayesian Stackelberg game, allowing the agent to appropriately weigh the different actions in randomization, as well as uncertainty over adversary type. ARMOR combines three key features: (i) It uses DOBSS, which uses mixed strategies to enable randomization; (ii) The mixed-initiative interface allows users to adjust or override the automated schedule based on their local constraints; (iii) It alerts the users if mixed-initiative overrides appear to degrade the overall desired randomization.

Intelligent Randomization In Scheduling (IRIS) was developed to schedule FAMS (Fed-

eral Air Marshal Service) on flights [29]. The game models which this system is based on are very large as there are tens of thousands of flight everyday and thousands of air marshals with complex scheduling constraints. IRIS is modeled as Stackelberg game, with FAMs (Federal Air Marshal) as leaders that commit to a flight coverage schedule and terrorists as followers that attempt to attack a flight. IRIS is currently deployed as part of the international scheduling practices of FAMS.

Game-theoretic Unpredictable and Randomly Deployed Security (GUARDS) uses a game theoretic approach for national scale security deployment for airport security [26]. The scheduling assistant has been delivered to the Transportation Security Administration (TSA) and is currently under evaluation and testing for scheduling practices at an undisclosed airport. Previous applications like ARMOR and IRIS were focused on one-off tailored applications and one security activity (e.g., canine patrol or checkpoints) per application. GUARDS focuses on reasoning about hundreds of heterogeneous security activities and diverse potential threats.

Port Resilience Operational/Tactical Enforcement to Combat Terrorism (PROTECT) is a game-theoretic system deployed by the United States Coast Guard (USCG) in the ports of Boston, Los Angeles and New York for scheduling their patrols [28]. PROTECT is modeled based on attacker-defender Stackelberg game model and avoids the assumption of perfect adversary rationality like previous works and relies on a Quantal Response (QR) model of the adversary's behavior.

Game theory is also used for applications in cybersecurity [3, 22]. Using game-theoretic concepts, a generic model has been developed for distributed Intrusion Detective Systems (IDS) with network sensors [3]. In this work, two platform-independent game-theoretic schemes have been proposed which addresses basic security trade-offs. To protect network from possible attackers, there have been game-theoretic investigations of the effects of deception on the interactions between attacker and defender of a computer network [7]. This is an incomplete-information non-cooperative signaling game where defender can employ camouflage by either disguising a normal system as a honeypot, or by disguising a honeypot

as a normal system.

There are also applications of game theory for patrolling strategies for robots and unmanned vehicles and for network security [3, 22, 30]. Much of these papers are computational in nature, and progress on security games has been driven by many algorithms that made solutions more robust and applicable to real-life scenarios.

(a) A vehicle checkpoint at LAX

(b) IRIS scheduling system

Figure 2.1: Applications of security games

2.2 Modeling uncertainty in security games

Game theory is becoming a vital framework for reasoning about real world security problems including critical infrastructure protection. The game models for these applications are based on expert analysis and historical data which includes activities and possible characteristics of terrorists. So it is quite natural to represent uncertainty over these parameters. One of the major challenges in developing security game model is handling uncertainty. To model uncertainty in games, the Bayesian game model was proposed by John Harsanyi [11], which defines nature as one of the players in the game. Harsanyi developed a theory for the

analysis of games with incomplete informations, where players are uncertain about some important parameters of the game situation such as the payoff functions, the strategies of other players, etc.

Bayesian games are also useful to represent uncertainty in security games. Bayesian Stackelberg games allow us to explicitly model players as types, where each type can have its own preference. However finding a Stackelberg equilibrium in Bayesian game is NP-hard [8]. There are several works on problems of robustness in security games, most of which adopt a Bayesian framework for reasoning about uncertainty [12, 16, 23, 24, 31]. Some notable works on Bayesian games applied to security games are DOBSS [23], HBGS [12], IBGS [16] and HUNTER [31], which will be described in turn.

Decomposed Optimal Bayesian Stackelberg Solver (DOBSS) analyzes the entire Bayesian game at once without using the Harsanyi transformation by using a mixed-integer linear program, which optimizes against each adversary type independently while keeping the leader strategy fixed across all types. However, this method fails to scale up beyond 10 types even for 20 actions for the players, or beyond 30 actions for just 5 follower types.

Hierarchical Bayesian solver for General Stackelberg games (HBGS) is a much faster algorithm for finite Bayesian Stackelberg games. This algorithm was used in the original ARMOR application at the LAX airport. This technique is based on heirarchical decomposition and branch and bound search over the follower type space. But it is still too slow to scale to domains such as FAMS (Federal Air Marshals Service) with thousands of actions and restricting the model to small number of attacker types can lead to poor solution quality.

Handling UNcerTainty Efficiently using Relaxation (HUNTER) provides an unified approach to handle uncertainty not over only discrete follower types but also other continuously distributed real world uncertainty including the leader's execution error, the follower's observation error and continuous payoff uncertainty. HUNTER continues the trend of speed improvements and is even faster than HBGS, and is currently the fastest known algorithm for finite Bayesian Stackelberg games. The main advantage of HUNTER is that it handles

both type of uncertainty simultaneously by scaling up Bayesian Stackelberg games and it performs better than other competing algorithms.

Infinite Bayesian Stackelberg games (IBGS) is able to solve large game models with infinite number of attacker types and payoff uncertainty which are represented by distributions [16]. DOBSS and HBGS are used to solve games with finite number of attacker types. However, restricting models to a small number of attacker types leads to poor solution quality. To solve incomplete information games there are several approximate methods, among which Greedy Monte Carlo (GMC) has the best performance in solution quality and scalability. It uses a greedy heuristics for allocating defender resources with very fast method for updating the attacker response function estimated using Monte Carlo Sampling.

There are also other specifics examples of infinite Bayesian games that have been solved analytically, including many types of auctions [19]. However there are few works for solving large and infinite Bayesian games like IBGS. Monte Carlo Sampling approaches similar to those used in IBGS have been applied to some kind of auctions [6]. In addition, the literature on stochastic choice [20, 21] studies problems that are simplified versions of the choice problem attackers face in our model.

2.3 Worst-Case Approaches to Robustness

An alternative to Bayesian games that has been developed recently is robust equilibrium [1], which proposes a distribution free model and takes a worst-case approach inspired by robust optimization literature for incomplete information games. Robust optimization is a modeling methodology, combined with other computational tools, to process optimization problem in which data are uncertain. For robustness against human decision-makers [24], the most closely related model for security games is BRASS (Bounded Rationality Assumption in Stackelberg Solver), which was introduced by Pita et al [26]. This model assumes that the follower is boundedly rational and may not be able to maximize utility strictly. The follower selects one response from multiple optimal responses and the robust approach

is to assume that the follower could choose the response that provides the leader the worst reward value.

Current algorithms for Stackelberg games are based on two fundamental assumptions. The first assumption is that the followers act rationally to maximize the utility, and the second, if the follower faces a tie in its strategies' rewards, it will break tie in favor of the leader, choosing a strategy that gives higher rewards to the leader. However, leader may face followers who fail to response optimally in real world domains. This may be caused by follower's bounded rationality or uncertainty regarding leader's strategy, and there is no prior probability distributions available for this follower response uncertainty.

To overcome this situation, decision makers optimize against the worst outcome over the uncertainty [1]. In solution methods like BRASS, leaders make robust decision by considering that the bounded rational follower could choose a strategy from its range of responses that degrades the leader's rewards the most or that he could choose a strategy that is based on limited observations. This approach is in contrast with standard robust optimization methodology in that it makes prediction about how and why the human adversary's response will deviate and robustly guards against those predictions as opposed to considering arbitrary deviations in the responses. BRASS addresses the uncertainty that may arise from human imprecision in choosing the expected optimal strategy due to bounded rationality.

2.4 Interval Computation

Interval computing is a board field in which rigorous mathematics is associated with scientific computing. This methods was first developed by mathematicians as an approach to put bounds on rounding errors and measurement errors in order to produce reliable results in mathematical computations. Interval arithmetic or computation can be used in various areas (e.g., set inversion, motion planning), in order to treat estimates for which no exact numerical values can be stated [13]. Applications based on interval computa-

tion are gaining popularity in quality control, economics, computer graphics and computer aided design, quantum mechanics as well as other fundamental ideas [14]. Some C++ and Fortan compiler support interval arithmetic directly by handling interval data types and suitable operation such as language extension. Many software packages allow development of numerical analysis using interval arithmetic [18].

Chapter 3

Security Games

In this chapter the ideas behind security games and interval security games are described.

3.1 Formal Definition of Security Games

In this section the basic security game model is introduced first and then this model is extended to include interval uncertainty about the attacker's payoffs. A security game consists of two players, a *defender* Θ and an *attacker* Ψ. The defender is protecting a set of *targets* $T = \{t_1, \ldots, t_n\}$ (e.g., airport terminals or computer servers) from attacks using a limited number of resources, with the number of available resources denoted by m. It is assumed that all resources are identical and can be used to protect any target. The set of pure strategies for the attacker, denoted by Σ_Ψ, correspond to actions of attacking exactly one target from T. Each pure strategy for the defender corresponds to a subset of targets from T with size less than or equal to m.

In the model there are two kinds of payoff values: values for covered targets and values for uncovered targets. Table 3.2 contains necessary notations and their meanings. An attack on a covered target is "unsuccessful" and an attack on an uncovered target is "successful". In a security game it is also assumed that $U^c_\Theta(t) \geq U^u_\Theta(t)$ and $U^u_\Psi(t) \geq U^c_\Psi(t)$ for all $t \in T$ where $U^c_\Theta(t)$ and $U^u_\Theta(t)$ denote defender's covered and uncovered payoffs and $U^c_\Psi(t)$ and $U^u_\Psi(t)$ denote attaker's covered and uncovered payoffs respectively. This is analogous to a real-world security scenario. Whenever a target is uncovered and if there is an attack on that target, the attacker gets higher payoff and defender gets a lower or negative payoff. On the other hand if the target is covered and there is an attack, defender gets higher payoff

Table 3.1: A sample game

	c	d
a	$(2, 1)$	$(4, 0)$
b	$(1, 0)$	$(3, 2)$

and the attacker gets a lower one.

The interaction between the attacker and the defender is modeled as a Stackelberg game. The main idea behind the Stackelberg game is that the follower can observe strategies of the leader and act accordingly. In a security game scenario, the attacker (or follower) observes the defender's (or leader's) strategy and then takes action. The attacker plays a pure strategy while the defender plays a mixed strategy. The next paragraph describes why playing a mixed strategy will give the defender a better payoff. Since the attacker sees the defender's strategy, there always exists a pure strategy best response, so the attacker does not need a mixed strategy.

In the game in Table 3.1, the defender is a row player and the attacker is a column player. The defender has a choice of playing a or b and the attacker has a choice of playing c or d. Each of the squares defines a possible outcome which contains each player's payoffs for that outcome. For example if the defender plays a and the attacker plays c, the defender will have a payoff of 2 and the attacker will have a payoff of 1.

If the defender opts for a pure strategy, he chooses a, as it ensures that minimum and maximum defender's payoff is greater than the other choice b. The attacker will choose c based on the defender's decision a. In the end the defender has a payoff of 2. Now, suppose the defender opts for a mixed strategy such that instead of choosing a or b, he can choose each of them 50% of the time. In that case the attacker will choose his best response based on the defender's action. If the attacker plays c observing the defender's actions, he receives a payoff of $0.5(1 \cdot 0.5 + 0 \cdot 0.5)$ and by playing d he receives 1 $(0 \cdot 0.5 + 2 \cdot 0.5)$. As the attacker gets higher payoff by playing d, he will choose action d and the defender

will receive a payoff of 3.5 ($0.5 \cdot 4 + 0.5 \cdot 3 = 3.5$). On the other hand, if the defender plays a pure strategy, he receives a payoff of 2. So by playing a mixed strategy the defender receives a better reward value. So the defender first commits to a mixed strategy δ_Θ that is a probability distribution over the pure strategies from Σ_Θ. The attacker then observes this mixed strategy δ_Θ, and chooses a best response strategy from Σ_Ψ that gives the attacker the maximum possible payoff.

The defender's mixed strategies are represented as *coverage vectors* which gives the probability that there is a defender resource assigned to each target. These probabilities for each target t_i are denoted by c_i, with $\sum_{i=1}^{n} c_i = m$, and the full vector of probabilities is denoted by C. These are the targets that the defender chooses to protect.

3.2 Interval Security Games

One of the hardest parts of solving security game problems is generating the game models which can be done based on historical data, intelligence, and opinions from expert analysts. However, precise estimation of these parameters is difficult in practice. There are two general approaches to generate these game models: Bayesian game models and Interval game models. Both of these game models use uncertainty on payoff functions. Bayesian game models use distributions over payoff functions.

The game model presented in this thesis is named *Interval Security Game* model, which extends the model of standard security game. In this model the defender has uncertainty about the attacker's payoffs in the form of intervals, and both the defenders and the attackers know their own payoffs. As the attacker can directly observe the defender's strategies, the attacker does not need to predict defender's strategy. This model replaces the attacker's covered and uncovered payoffs with ranges of payoffs as seen in Table 3.3. The defender only knows that the attacker's payoffs are in these ranges, and must predict the attacker response based on this. The solution methods used previously, including Bayesian Stackelberg equilibrium, can't be used in ISG model, as the defender does not know the

Table 3.2: Notations used in security games

$T = \{t_1, \ldots, t_n\}$	Set of Targets
t_i	i-th target i
c_i	Coverage for target i
C	Coverage Vector for all targets
m	Total number of resources
Θ	A defender
Ψ	An attacker
$U^u_\Theta(t)$	Defender's payoff for attack on uncovered target
$U^c_\Theta(t)$	Defender's payoff for attack on covered target
$U^u_\Psi(t)$	Attacker's payoff for attack on uncovered target
$U^c_\Psi(t)$	Attacker's payoff for attack on covered target
R	Maximum of the minimum expected payoff for attackers among all targets
$\Lambda(C)$	Potential attack set
δ_Θ	Set of defender's mixed strategies
Σ_Ψ	Set of attacker's pure strategies

Table 3.3: Notations of payoffs used in *Interval Security Game* models

$U_{\Psi}^{u,\min}(t)$	Attacker's minimum payoff for attacking an uncovered target t
$U_{\Psi}^{u,\max}(t)$	Attacker's maximum payoff for attacking an uncovered target t
$U_{\Psi}^{c,\min}(t)$	Attacker's minimum payoff for attacking a covered target t
$U_{\Psi}^{c,\max}(t)$	Attacker's maximum payoff for attacking a covered target t
$U_{\Theta}^{u}(t)$	Defender's payoff for attack on uncovered target
$U_{\Theta}^{c}(t)$	Defender's payoff for attack on covered target
$v^{\max}(t_i)$	Attacker's maximum expected payoff for target i
$v^{\min}(t_i)$	Attacker's minimum expected payoff for target i
d_i	Defender's expected payoff for target i

distributions of the attacker's payoff, but only the intervals. Instead the literature of robust optimization is followed and the defender's goal in this framework is to find a coverage vector, C, that maximizes the defender's worst-case payoff over all of the possible ways that the attacker payoffs could be chosen from the intervals.

Previously defined models use two parameters to represent the attacker's payoff, as seen in Table 3.2. For each target t, $U_{\Psi}^{u}(t)$ and $U_{\Psi}^{c}(t)$ are used to represent the attacker's uncovered and covered payoff respectively for that target. In the new model, each of the targets has a maximum and minimum payoff for the attacker's covered and uncovered payoffs, represented by these four notations, $U_{\Psi}^{u,\max}(t)$ and $U_{\Psi}^{u,\min}(t)$ for the uncovered case, and $U_{\Psi}^{c,\max}(t)$ and $U_{\Psi}^{c,\min}(t)$ for the covered case.

As the model contains the attacker's minimum and maximum payoff for each of the targets, the attacker's maximum and minimum *expected* payoff for each of the target can be calculated given c_i, which is the coverage probability for that target.

$$v^{\max}(t_i) = c_i \cdot U_{\Psi}^{c,\max}(t_i) + (1 - c_i) \cdot U_{\Psi}^{u,\max}(t_i) \tag{3.1}$$

$$v^{\min}(t_i) = c_i \cdot U_{\Psi}^{c,\min}(t_i) + (1 - c_i) \cdot U_{\Psi}^{u,\min}(t_i). \tag{3.2}$$

The defender's expected payoff for each target is:

$$d_i = c_i \cdot U^c_\Theta(t_i) + (1 - c_i) \cdot U^u_\Theta(t_i). \tag{3.3}$$

For a given coverage vector C, the attacker can guarantee a payoff of at least the maximum of the minimum values over all targets; let us denote this value by $R = \max_{t_i} v^{\min}(t_i)$. Given the value of R, the targets that could be attacked for some realization of the payoff values can be identified. Any target t_i with a maximum expected value $v^{\max}(t_i) \geq R$ could be the best target for the attacker to attack. To see this, suppose that the the attacker's payoff for t_i is the maximum value in the interval, and the payoffs for all other targets is are the minimal values, so that the best possible value for attacking any target other than t_i is R. Therefore, the potential attack set $\Lambda(C)$ is defined as:

$$\Lambda(C) = \{t_i : v^{max}(t_i) \geq R\} \tag{3.4}$$

The defender's objective is to find out a strategy C to maximize the worst-case payoff over all of the targets in the potential attack set:

$$\max_C \left(\min_{t_i \in \Lambda(C)} d_i \right) \tag{3.5}$$

In the above equation: $\min_{t_i \in \Lambda(C)} d_i$ defines the target which is in potential attack set and has minimum expected payoff for defender. This term is described as worst-case payoff. In this algorithm the objective is to find out a coverage vector which will maximize this worst case payoff.

This problem cannot be solved efficiently using linear programming because the set of targets $t_i \in \Lambda(C)$ depends on C. This can be solved using a mixed-integer program (MIP) which is a slightly generalized version of the MIP used for BRASS **??**, which runs slowly on general problems. Therefore a faster method was developed to solve this problem.

Chapter 4

Interval Security Games Algorithm

In this chapter, the basic idea behind the interval security game algorithm is explained first and then the next section presents how the algorithm is formulated.

4.1 Idea Behind ISG

The main idea for finding a coverage strategy with the desired properties is to transform the optimization problem into a series of feasibility problems. As the number of defenders (m) is increased, a better coverage vector can be achieved which will lead to better expected payoffs for the defender. This observation is used to formulate the problem as a binary search in the space of defender expected payoffs. In each iteration, it is checked whether the defender payoff at the midpoint of the search space is feasible or not, given that there are limited number of defenders. If it is not, the left half of the search space is checked. Otherwise, a similar check is done on the right half of the search space.

In the binary search approach, the problem is analyzed to determine whether a given defender payoff, which is denoted by D^*, is feasible or not given the resources available, m. Since worst-case outcomes need to be ensured, it needs to be guaranteed that the defender will achieve *at least* D^* for any attacker payoffs in the known intervals. For D^* to be guaranteed by a particular coverage strategy C, one of the following conditions must hold.

1. The target is in the potential attack set $\Lambda(C)$, but the defender's expected payoff for attacking the target is greater than D^*

2. The target is not in the potential attack set $\Lambda(C)$.

The coverage required on each target can be calculated to satisfy Condition 1 for each target (if it is in Λ) from the equation for the defender's payoff. The meanings of the notations used in this chapter can be found in Chapter 3 on Table 3.2. If a target has the full coverage (coverage probability of 1), the defender's expected payoff for that target will be $U^c_\Theta(t)$. On the other hand, if a target does not have the full coverage, the defender's expected payoff for that target can be calcuated by multiplying $U^c_\Theta(t)$ with coverage probability, and multiplying $U^u_\Theta(t)$ with the probability of the target being uncovered and summing them up. If the defender wants to maintain at least D^* for target t_i with coverage c^1_i, the following equation must hold.

$$
\begin{aligned}
& D^* = c^1_i \cdot U^c_\Theta(t_i) + (1 - c^1_i) \cdot U^u_\Theta(t_i) \\
\Rightarrow \quad & D^* = c^1_i \cdot U^c_\Theta(t_i) + U^u_\Theta(t_i) - c^1_i \cdot U^u_\Theta(t_i) \\
\Rightarrow \quad & D^* = c^1_i \cdot (U^c_\Theta(t_i) - U^u_\Theta(t_i)) + U^u_\Theta(t_i) \\
\Rightarrow \quad & c^1_i \cdot (U^c_\Theta(t_i) - U^u_\Theta(t_i)) = D^* - U^u_\Theta(t_i) \\
\Rightarrow \quad & c^1_i = \max\left(0, \frac{D^* - U^u_\Theta(t_i)}{U^c_\Theta(t_i) - U^u_\Theta(t_i)}\right). \qquad (4.1)
\end{aligned}
$$

This equation makes sure that if coverage c^1_i is maintained, it will guarantee payoff D^*. It is also noticeable that the term c^1_i is dependent on $D^* - U^u_\Theta(t_i)$. Let's assume that we want to make sure a target will guarantee a defender's payoff of 40. It has covered and uncovered defender's payoff 80 and 30 respectively. So the right term of above equation gives us the value $((40 - 30)/(80 - 30)) = 0.2$. If a target has coverage probability of at least 0.2, the defender's expected payoff for that target is 40 or greater.

For a given value of R, the minimum coverage can be calculated that would be required on each target t_i so that the target is *not* in Λ, which requires that the maximum possible expected payoff for attacking target t_i is less than R. The attacker's expected payoff for attacking a target can be calculated using the attacker's covered and uncovered payoff for attacking that target and the coverage probability. As the target is to find out the minimum coverage required for a target not to be in the attack set, the attacker's maximum payoff for that target will be used. The minimum coverage c^2_i is calculated as follows, using the

maximum attacker payoffs from the possible intervals:

$$
\begin{aligned}
& R = c_i^2 \cdot U_\Psi^{c,max}(t_i) + (1 - c_i^2) \cdot U_\Psi^{u,max}(t_i) \\
\Rightarrow \quad & R = c_i^2 \cdot U_\Psi^{c,max}(t_i) + U_\Psi^{u,max}(t_i) - c_i^2 \cdot U_\Psi^{u,max}(t_i) \\
\Rightarrow \quad & R = c_i^2 \cdot (U_\Psi^{c,max}(t_i) - U_\Psi^{u,max}(t_i)) + U_\Psi^{u,max}(t_i) \\
\Rightarrow \quad & c_i^2 \cdot (U_\Psi^{c,max}(t_i) - U_\Psi^{u,max}(t_i)) = R - U_\Psi^{u,max}(t_i) \\
\Rightarrow \quad & c_i^2 = \max\left(0, \frac{R - U_\Psi^{u,max}(t_i)}{U_\Psi^{c,max}(t_i) - U_\Psi^{u,max}(t_i)}\right). \qquad (4.2)
\end{aligned}
$$

By summing values of c_i^1 for targets in Λ and c_i^2 for the remaining targets, the minimum coverage can be found which is required to guarantee D^* for this potential attack set. A possible brute force approach of finding c_i^1 and c_i^2 can be applied. But, the number of such sets is exponential in the number of targets, so this direct approach is highly inefficient. To avoid this problem, another observation is made that allows to more efficiently explore candidate solutions. For every possible set Λ there must be some target, which is labeled as $\hat{t}$ that has the maximum minimum expected payoff, R. This is the target that defines the value of R. Since there are only n targets, each target can be tested as a candidate for $\hat{t}$, and construct a coverage vector that meets the necessary constraints using minimal resources under this assumption. If the solution is feasible for any one of the n targets that are candidates for $\hat{t}$, then the value of D^* is feasible. By taking the minimum of c_i^1 and c_i^2 to calculate the coverage vector, it is made sure that minimum coverage is being used to guarantee D^* or to confirm that that target is not in the attack set.

There is one final condition that must be met for each target for the initial assumption to hold. It must be the case that the target assumed to be $\hat{t}$ actually has the maximum minimum expected payoff. This means that all targets other than $\hat{t}$ must have a minimum attacker payoff which is less than the value of R. This is guaranteed by adding one additional constraint on the coverage probability assigned to each of the targets. This constraint states that the minimum attacker payoff for the target is less than the calculated value of R.

$$R = c_i^3 \cdot U_\Psi^{c,min}(t_i) + (1 - c_i^3) \cdot U_\Psi^{u,min}(t_i)$$

$$
\begin{aligned}
\Rightarrow \quad & R = c_i^3 \cdot U_\Psi^{c,min}(t_i) + U_\Psi^{u,min}(t_i) - c_i^3 \cdot U_\Psi^{u,min}(t_i) \\
\Rightarrow \quad & R = c_i^3 \cdot (U_\Psi^{c,min}(t_i) - U_\Psi^{u,min}(t_i)) + U_\Psi^{u,min}(t_i) \\
\Rightarrow \quad & c_i^3 \cdot (U_\Psi^{c,min}(t_i) - U_\Psi^{u,min}(t_i)) = R - U_\Psi^{u,min}(t_i) \\
\Rightarrow \quad & c_i^3 = \max\left(0, \frac{R - U_\Psi^{u,min}(t_i)}{U_\Psi^{c,min}(t_i) - U_\Psi^{u,min}(t_i)}\right) \\
\Rightarrow \quad & c_i^3 = \max\left(0, \frac{U_\Psi^{u,min}(t_i) - R}{U_\Psi^{u,min}(t_i) - U_\Psi^{c,min}(t_i)}\right). \qquad (4.3)
\end{aligned}
$$

The value of c_i^3 increases monotonically as R decreases. The final calculation for the minimum coverage that must be placed on each target is $\max(c_i^3, \min(c_i^1, c_i^2))$. These coverages over all the targets are summed up and compared with the available resources m to determine whether this selection of $\hat{t}$ yields a feasible solution or not. If this selection does not yield a feasible solution, the algorithm continues testing the other possible targets as selections for $\hat{t}$. As soon as a feasible solution is found, the subroutine terminates and the binary search continues.

4.2 ISG Algorithm

The ISG algorithm consists of two parts. Algorithm 1 implements binary search in the space of possible defender payoffs, as described in the previous section. The search continues until the difference between the maximum and the minimum possible defender's payoff falls below ϵ, which is the error tolerance parameter for the binary search. By considering a smaller value of ϵ, a better defender's payoff can be achieved. However, the solution time will be higher than the one with larger value of ϵ.

The feasibility check is presented in Algorithm 2 where the objective is to justify if the defender's payoff D^* is possible or not. If the feasibility check algorithm can generate a solution where the required resources is less than or equal to m, then D^* is possible. The strategy is to divide the search into n possible cases, each of which corresponds to a different assumption about which of the targets will have the maximum minimum expected payoff, R. The algorithm iterates through each possible choice of t_i as a candidate for this

Algorithm 1 ISG Solver

for all $t_i \in T$ **do**

$c_i \leftarrow 0$

end for

$maxPayoff \leftarrow \max_{t_i \in T} U^c_\Theta(t_i)$

$minPayoff \leftarrow \min_{t_i \in T} U^u_\Theta(t_i)$

while $maxPayoff - minPayoff > \epsilon$ **do**

$midPoint \leftarrow (maxPayoff - minPayoff)/2$

if feasibilityCheck($midPoint$, m, C) **then**

$minPayoff \leftarrow MidPoint$

else

$maxPayoff \leftarrow MidPoint$

end if

end while

return C

special target $\hat{t}$, which was defined earlier. For each of these cases the algorithm constructs a coverage vector using minimal coverage probability that guarantees the defender's payoff D^*, based on conditions 1 and 2 stated above. Once $\hat{t}$ is selected, the value of $c^1_{\hat{t}}$ is calculated which is necessary to ensure D^* and used to calculate the value of R. Using that R, c^2_i for all other targets are found out. Finally another constraint is evaluated which ensures that $\hat{t}$ has a minimum expected payoff which is maximum among all the minimum expected payoffs of all targets.

The worst-case complexity of the algorithm is $O(n^2 \cdot \log(1/\epsilon))$. Each feasibility check requires one iteration to test each target as $\hat{t}$, and each iteration does several constant-time operations on each target to determine the minimal coverage. Therefore, the feasibility check has complexity $O(n^2)$. Binary search requires $O(\log(1/\epsilon))$ iterations to converge within ϵ, giving the overall complexity of $O(n^2 \cdot \log(1/\epsilon))$.

Algorithm 2 feasibilityCheck

for all $t_i \in T$ **do**

$c_i^1 \leftarrow \max(0, 1 - \frac{midPoint - U_{\Theta}^{u}(t_i)}{U_{\Theta}^{c}(t_i) - U_{\Theta}^{u}(t_i)})$

end for

for all $t_i \in T$ **do**

$totalCov \leftarrow c_i^1$

$c_i \leftarrow c_i^1$

if $c_i > 1$ **then**

$GOTO\ next\ t_i$

end if

$R \leftarrow (c_i^1 \cdot U_{\Psi}^{c,min}(t_i)) + ((1 - c_i^1) \cdot U_{\Psi}^{u,min}(t_i)) - \epsilon'$

for all $t_j \in \{T \backslash t_i\}$ **do**

$c_j^2 \leftarrow \max\left(0, 1 - \frac{R - U_{\Psi}^{u,max}(t_j)}{U_{\Psi}^{c,max}(t_j) - U_{\Psi}^{u,max}(t_j)}\right)$

$c_j^3 \leftarrow \max(0, 1 - \frac{R - U_{\Psi}^{u,min}(t_j)}{U_{\Psi}^{c,min}(t_j) - U_{\Psi}^{u,min}(t_j)})$

$minCov \leftarrow max(c_i^3, \min(c_i^1, c_i^2))$

if $minCov < 0$ **or** $minCov > 1$ **then**

$GOTO\ next\ t_i$

end if

$totalCov \leftarrow totalCov + minCov$

$c_j \leftarrow minCov$

end for

if $totalCov \leq m$ **then**

return TRUE, C

end if

end for

return FALSE

Chapter 5

Security Games with Distributional Uncertainty

As described previously, one of the major challenges in solving security games is generating the game model, and there are two ways of doing it, one with the interval uncertainty and another with the distributional uncertainty. In this chapter, I describe briefly the formulation of distributional uncertainty and how it was used in previous solution methods. I also show the techniques of transforming the distributional game into interval game.

5.1 Distributional Game Model

In the distributional game model, distributions are used instead of intervals to represent uncertainty. The distributional security games (DSG) model introduced by Kiekintveld et al [15] uses this approach and presents various approximation algorithm for computing solutions to the resulting infinite Bayesian Stackleberg games. DSG model contains more information as it has access to distributional information, but there are two significant drawbacks.

1. The models are more problematic to define by modeler.
2. These distributional models are computationally challenging and no exact algorithms are known to compute the exact optimal solution [16].

Compared to distributional security games, interval game models are easy to generate and use. However, in this thesis, I present a technique to convert that distributional

game model to interval game model. After the conversion, the interval algorithm is used to solve the converted model. In the experimental evaluation section I compared this approximations with the Greedy Monte Carlo (GMC) method which operates directly on a distributional game.

In distributional security games payoffs are represented as continuous probability density functions (e.g., uniform or Gaussian distributions). The game becomes an infinite Bayesian Stackleberg game with infinite number of attacker types. The following is the sequence of interactions:

1. The defender commits to a mixed strategy.

2. Nature chooses a random attacker type by drawing each payoff from the payoff distribution given in the model for that payoff.

3. The attacker observes the defender's mixed strategy.

4. The attacker plays the best-response that gives attacker the highest expected payoff, given the specific payoffs drawn by nature.

As seen in Figure 5.1, each of the target has different distributions for the attacker's payoff, based on that target being covered or uncovered by the defender. The defender knows only these distributions, but the attacker knows the specific payoffs drawn by nature. Attacker response function needs to be calculated first which returns the probabilities that each target will be attacked, given the distribution of the attacker's payoffs and coverage vector C.

Monte Carlo simulation can be used to estimate the attack probabilities on each target. In DSG, we generate one sample attacker payoff from payoff the distribution for each target and these payoff values are assigned to a sample attacker type. Using those payoff values we calculate the best-response for this attacker type against the coverage strategy C. Sampling a large number of types in this way is used to estimate the expected value of a coverage strategy for a DSG.

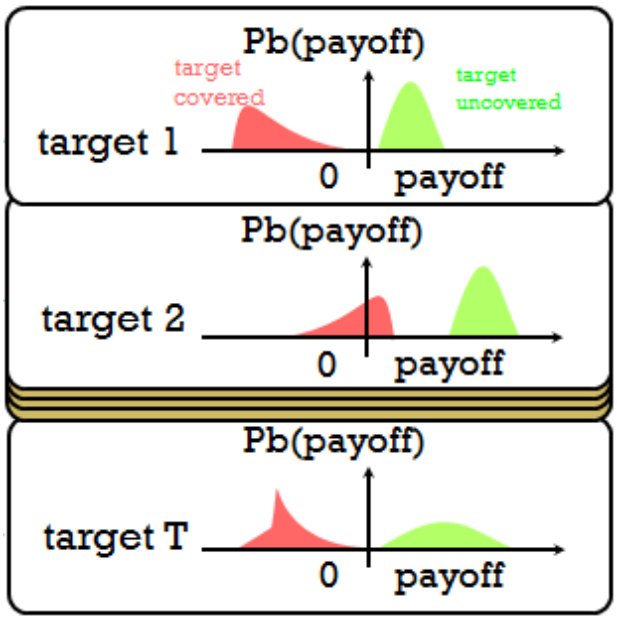

Figure 5.1: Distribution of attacker payoff

Greedy Monte Carlo (GMC), which is the best known method for solving distributional games [16], adopts a greedy heuristic for resource allocation and Monte Carlo sampling is used for updating the attacker's response function. The idea of this greedy heuristic is that initially all targets have *0* coverage probability assigned to the target. At each iteration, this method evaluates the prospect of adding some small increment (Δ) of coverage probability to each target. The algorithm computes the difference between the defender's expected payoff for the current coverage vector C and the new coverage vector that differs only in the coverage for a single target t such that $c'_t = c_t + \Delta$. The target with the maximum payoff gain for the defender is selected, Δ is added to the coverage for that target, and the algorithm proceeds to the next iteration. It terminates when all of the available resources have been allocated.

To apply our novel interval algorithm to distributional security games we need to transform the distributional payoffs to interval payoffs. We used a simple method that centers the interval around the mean of the distribution and determines the size of the interval

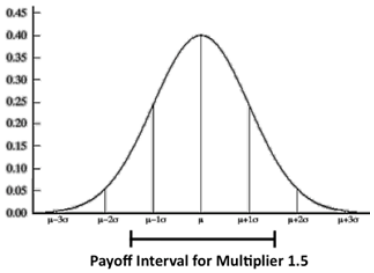

Figure 5.2: Interval payoff for a Gaussian distribution.

based on the standard deviation (σ) of the distribution and a multiplier, m. The multiplier is a parameter of the algorithm and allows us to have intervals that include a larger or smaller fraction of the possible payoff values in the distribution by adding or subtracting from the mean (μ) of the distribution. After applying this method we have the following two equations. Figure 5.2 explains this transformation. μ_1 and μ_2 represents "mean" for attacker's uncovered and covered payoff distribution respectively.

$$U_{\Psi}^{u,min}(t) = \mu_1 - (\sigma \cdot m) \tag{5.1}$$

$$U_{\Psi}^{u,max}(t) = \mu_1 + (\sigma \cdot m) \tag{5.2}$$

$$U_{\Psi}^{c,min}(t) = \mu_2 - (\sigma \cdot m) \tag{5.3}$$

$$U_{\Psi}^{c,max}(t) = \mu_2 + (\sigma \cdot m) \tag{5.4}$$

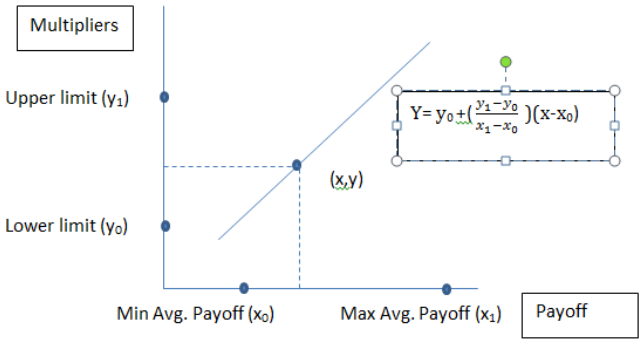

Figure 5.3: Linear Interpolation to find the value of multiplier m.

Additional experiments were performed to find out the value of the *multiplier*. For this purpose, two additional parameters were used, which are *modifiers* and *ranges*. Using the values of *ranges* and *modifiers*, the upper limits and the lower limits of the *multipliers* were calculated. As seen in the Figure 5.3, the x-axis contains minimum and maximum average attacker payoff and the y-axis contains the limits of the multipliers. For each of the targets, a different multiplier was calculated using the linear interpolation. Then the attacker's interval payoff for each target was calculated using the *multiplier* and the equations stated above. However, this set of experiments did not show substantial improvement over the basic version.

Chapter 6

Experimental Evaluation

Interval Security Games introduce a novel approach to represent uncertainty in security games. Experiments done on this algorithm show an improvement in the quality of solutions and solution time. I ran experiments on randomly generated sample games with various game parameters. The next two sections present experimental results on run-time and solution quality of ISG solver algorithm on interval security game, and the performance of this method to solve distributional security games.

6.1 Experiments on Interval Games

First the speed of ISG solver is tested against an exact Mixed Integer Programming (MIP) formulation based on BRASS [24]. Table 6.1 shows the game parameters used for this experiment. Games with various number of targets were generated and it was assumed that the number of defenders or security personnel is always 20% of the number of targets.

Table 6.1: Parameter settings for the experiments on Figure 1

Parameter	Values
Number of Sample Games	30
Defender Payoff for attack on uncovered target	random between -100 and 0
Attacker Payoff for attack on uncovered target	random between 0 and 100
Payoff for attack on covered target	0 (for both players)
Number of resources	20% of number of targets

The attacker payoffs were modified to be intervals by using the first value drawn from distribution of payoffs as the minimum value and and setting the maximum value by adding a uniform random value between 0 and 20. The tolerance settings control the accuracy of binary search. A higher tolerance means higher accuracy but takes more time compared with lower tolerance. Figure 6.1 presents results for the MIP which was solved using GNU Linear Programming Kit (GLPK version 4.36) [10] and ISG with three tolerance settings. Even with higher tolerance, ISG is much faster. Figure 6.2 shows ISG settings on much larger game. In this figure, a modest increase is visible in the solution time with increasing accuracy. Even for 10000 targets and the highest accuracy setting, ISG solves the game in half the time required by the MIP to solve games with only 300 targets.

Figure 6.3 shows experimental data based on the impact of interval uncertainty on solution quality under varying assumptions about the attacker strategy. Parameters listed in Table 6.2 were used for this experiment. The purpose of the experiment is to show how adding interval uncertainty decreases the best case payoff and improves against the worst case.

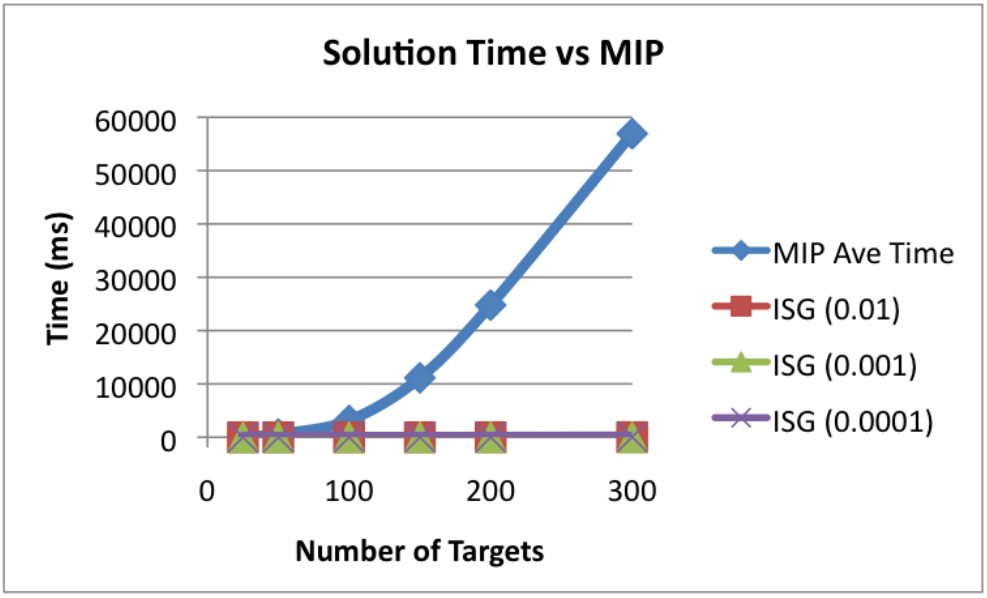

Figure 6.1: Comparison of solution time for ISG and the MIP solved using GLPK.

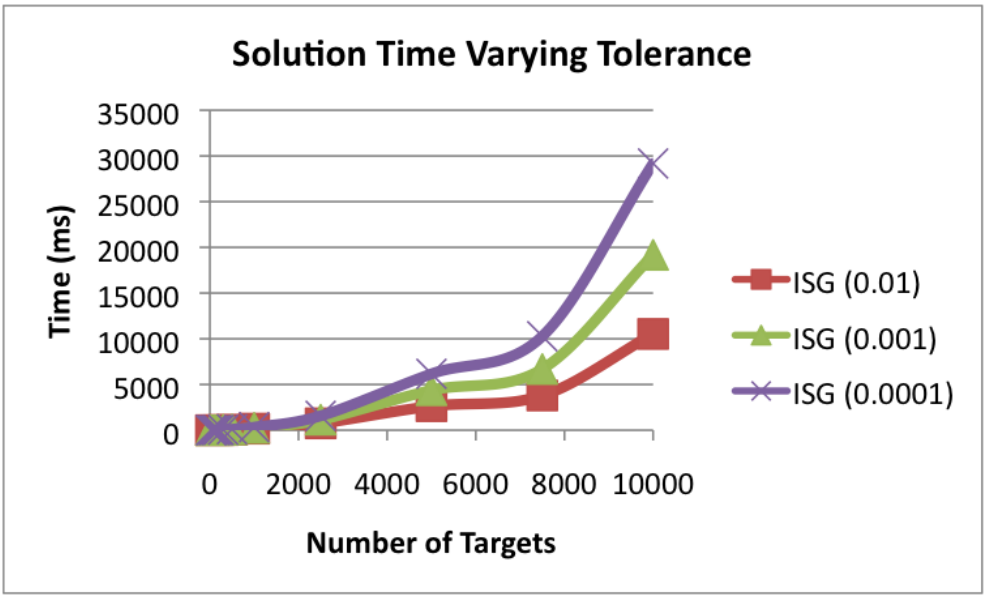

Figure 6.2: Impact of interval uncertainty on solution quality and robustness

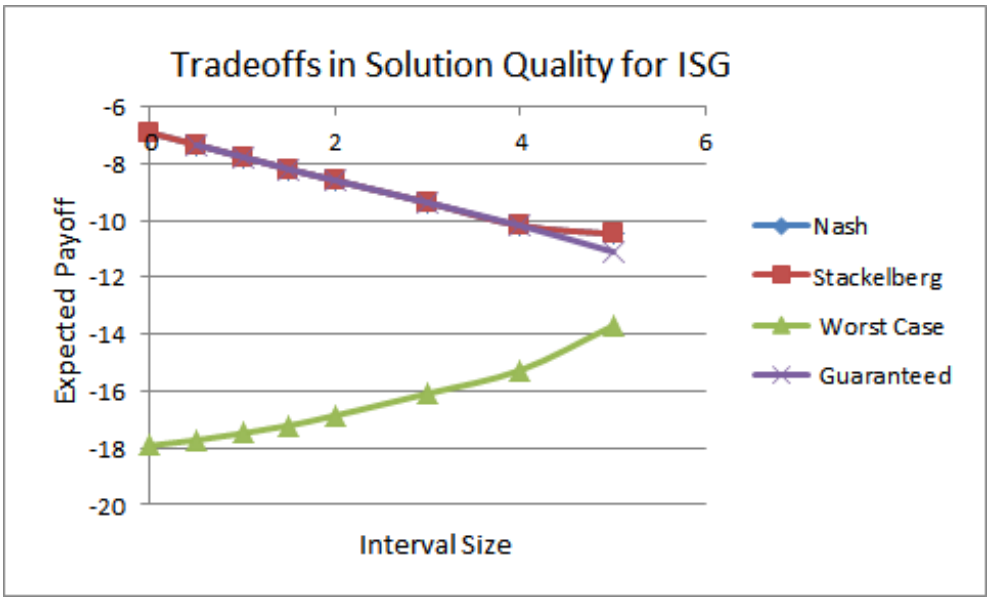

Figure 6.3: Runtime and solution quality analysis for ISG.

In Figure 6.3, the baseline case has no uncertainty. Increasing amount of interval uncertainty was added to the attacker's payoff and the resulting game was solved using the ISG solver. In this figure, the x-axis contains the interval sizes and the y-axis contains the

Table 6.2: Parameter settings for the experiments on Figure 6.3

Parameter	Values
Number of Sample Games	20
Defender Payoff for attack on uncovered target	random between -20 and -10
Attacker Payoff for attack on uncovered target	random between 10 and 20
Payoff for attack on covered target	0 (for both players)
Number of resources	20% of number of targets

defender's expected payoff. The four different lines represent four different assumptions about the attacker. The "Nash" attacker always plays the optimal attacker strategy computed in the case with no uncertainty (in this case, the Stackelberg equilibrium strategy is the same as the Nash strategy [17]). The "Stackelberg" attacker is able to observe the exact coverage strategy used in each case, and chooses a best-response, as in a Strong Stackelberg Equilibrium. The "Worst case" attacker always chooses the worst possible target for the defender, without regard to the attacker's own payoffs. Finally, the "Guaranteed" payoff is the payoff that the ISG method is able to guarantee against any rational attacker with payoffs that lie within the given intervals.

In Figure 6.3, there is a small decrease in the payoffs for the solutions to the interval games against the Nash and Stackelberg attackers. This is expected, and can be interpreted as the price of robustness. The advantage of the ISG method comes in the Guaranteed and Worst Case payoffs. There is an increasing trend in the worst-case payoffs for ISG, with the strongest results for very large intervals.

6.2 Experiments on Distributional Games

In this set of experiments, the performance of ISG is compared using the methodology for transforming distributional security games into approximate versions based on intervals to

Table 6.3: Parameter settings for the algorithms tested in the first experiment.

Parameter	Values
Number of sample Games	300
ISG Multipliers	0.6, 0.8, 1.0, 1.2, 1.4, 1.6, 1.8, 2.0
BRASS Epsilons	0.6, 0.8, 1.0, 1.2, 1.4, 1.6, 1.8, 2.0
GMC Low	increment 0.05, 1000 types
GMC High	increment 0.01, 10000 types

the best existing methods for DSG, Greedy Monte Carlo (GMC) [16] and BRASS [24]. Experiments were done on the same three classes of distributional games used by Kiekintveld et al. [16], games with Uniform distributions of payoffs, games with Gaussian distributions with fixed standard deviation (σ) for every payoff, and games with Gaussian distributions with varying standard deviations (σ), which we will call Gaussian Variable games. This set of experimental results evaluates the potential for ISG to be used as a fast approximation algorithm for distributional security games. Parameters used for this sets of experiments are described in Table 6.3.

The games are generated by first drawing random rewards and penalties for both players. All rewards are drawn from U[6;8], which defines a uniform distribution where minimum and maximum values are 6 and 8 respectively, and penalties are drawn from U[2;4]. Then distributions of the correct type for the attacker's payoff were generated, using the values drawn in the first stage as the mean. In uniform games, the length of the uniform interval was varied to increase or decrease the uncertainty. For Gaussian games, the standard deviation was varied, and all payoffs have the same amount of uncertainty. Gaussian variable games have a different standard deviation for each payoff distribution. These standard deviations are drawn from U[0;1] in the experiments.

In GMC there are two parameters that control the solution time and quality. These are the number of sample attacker types and size of the increment used in greedy allocation of

coverage probability. Solution quality improves with a larger number of types and smaller increment, but the solution time increases. That's why parameters were divided for GMC into "high" and "low" sections as found in Table 6.3.

The BRASS method has a parameter ϵ, degree of accuracy, which reflects how far attackers may be from choosing the optimal target. The parameters for ISG are "multipliers" and tolerance. Multiplier is used to generate the interval game from the DSG which is described in Chapter 5. The tolerance is fixed to 0.0001 in our experiments, since auxiliary experiments showed that the value of the tolerance has very small effect on the solution quality.

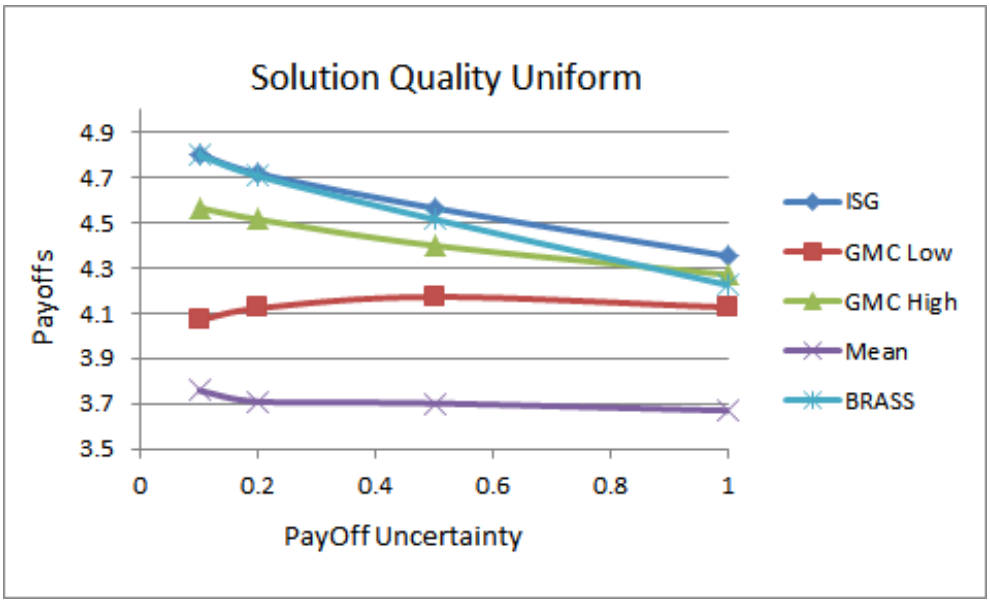

Figure 6.4: Solution quality results for small games with uniform attacker payoff distributions

In Figure 6.4, Figure 6.5 and Figure 6.6 all the experiments are done on small games (15 targets and 3 resources). These 3 graphs compare solution quality of various approximation algorithm on three types of distributional games: Uniform, Gaussian and Gaussian Variable. For the Uniform and Gaussian games the amount of payoff uncertainty was varied on the x-axis by varying the standard deviations of the attacker's payoff distributions.

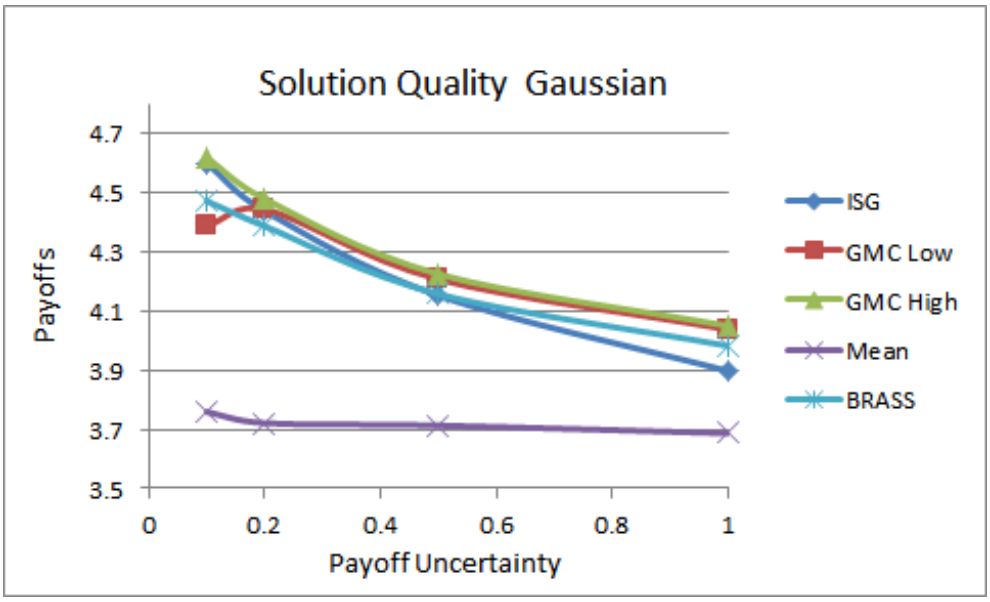

Figure 6.5: Solution quality results for small games with Gaussian attacker payoff distributions.

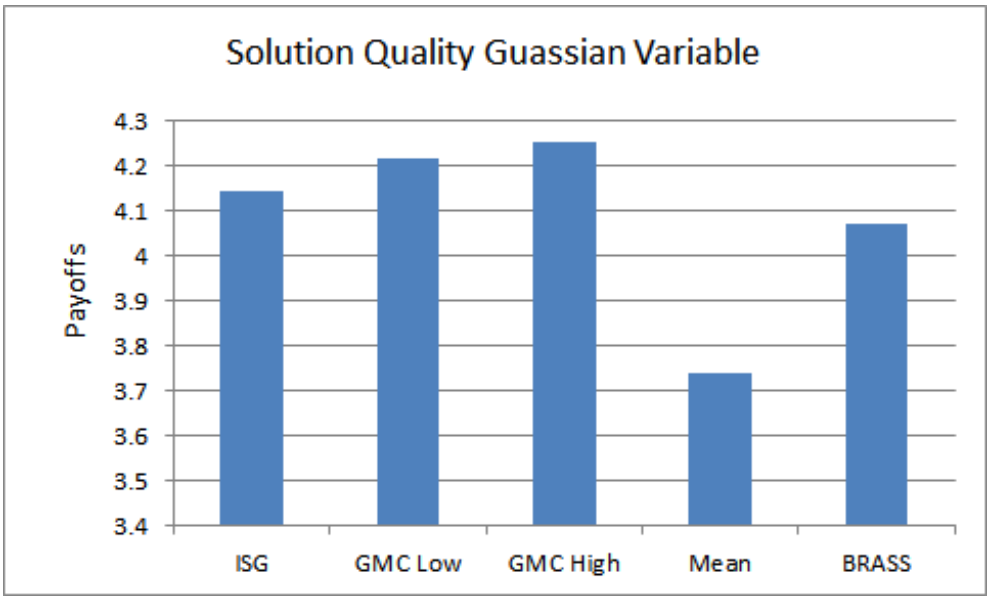

Figure 6.6: Solution quality results for small games with Gaussian variable attacker payoff distributions

For Gaussian Variable distributional games, a different parameter setting was used. All games have a different standard deviation for each payoff distribution. These standard deviations are drawn from uniform distribution U[0;1] in the experiments. In all cases, the defender's expected payoff is plotted on the y-axis. This is evaluated after the algorithms return solutions by using a very large number of Monte-Carlo sample types (100,000) to give a very accurate estimate of the expected payoff for the proposed coverage solution.

The experiment also includes a final baseline method called "Mean" that solves the game optimally under the assumption that the mean of the distribution is the exact payoff value (in other words, it ignores the uncertainty in payoffs and solves it as a standard security game). The mean baseline performs very poorly in all cases. For uniform games, ISG has the highest solution quality, followed closely by BRASS. For Gaussian and Gaussian variable games ISG performs slightly worse than the GMC methods, particularly when there is a large amount of uncertainty. However, the performance is still very competitive.

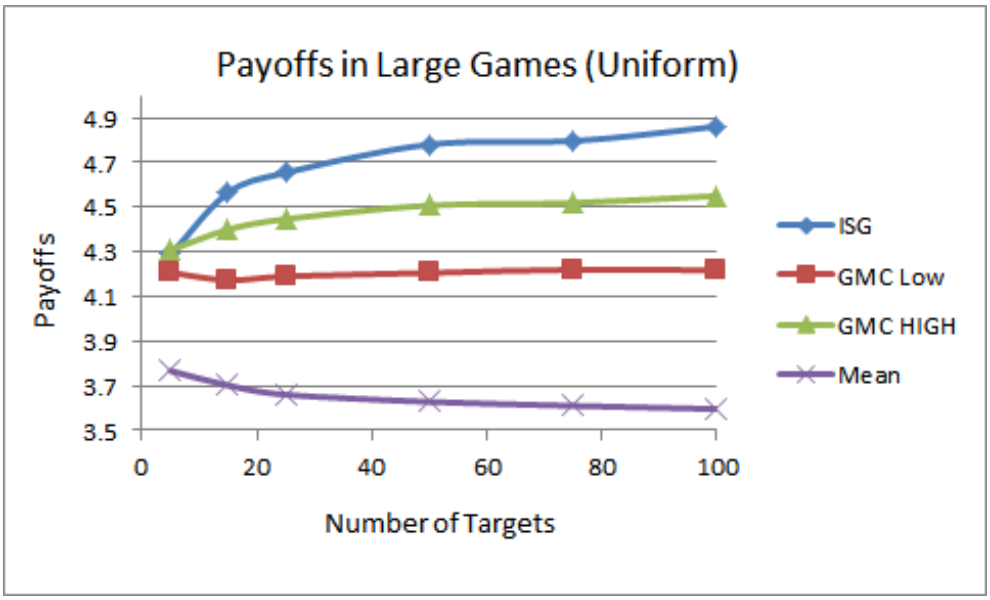

Figure 6.7: Solution quality results scaling to larger games for uniform attacker payoff distributions

Figure 6.7, Figure 6.8 and Figure 6.9 compare the methods for larger games (with higher

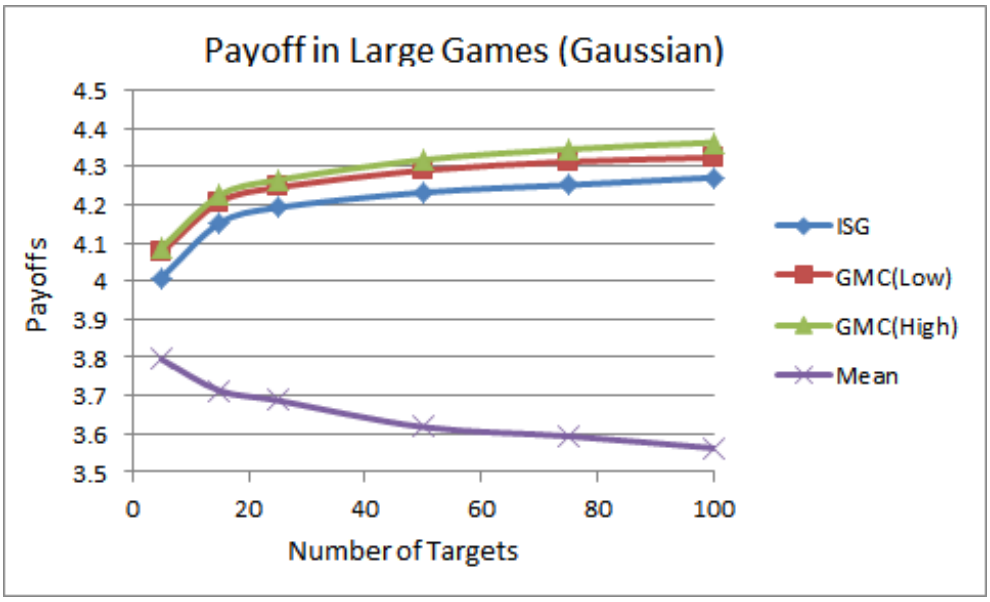

Figure 6.8: Solution quality results scaling to larger games for Gaussian attacker payoff distributions.

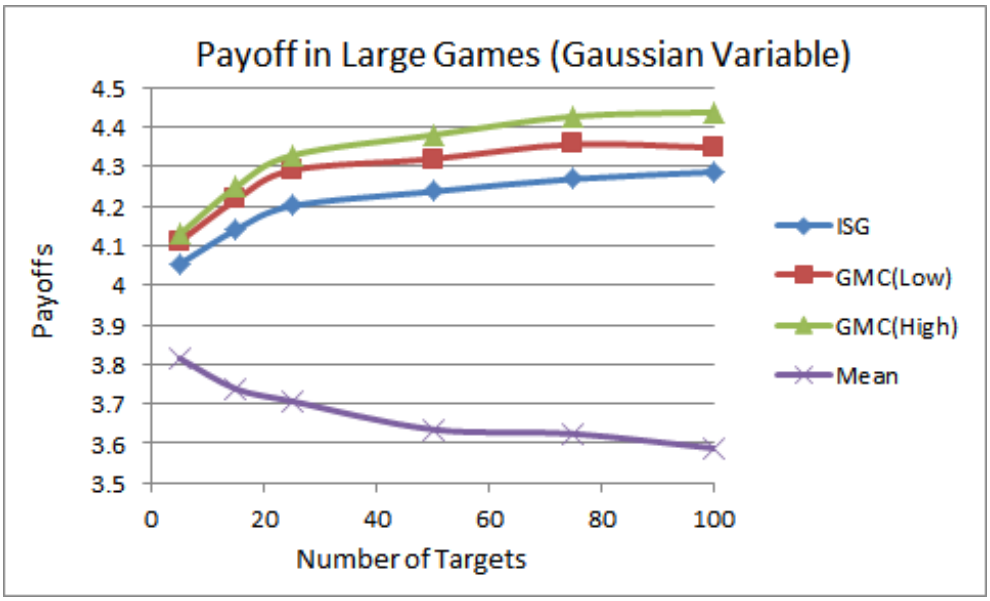

Figure 6.9: Solution quality results scaling to larger games for Gaussian variable attacker payoff distributions

number of targets). The standard deviation for uniform and Gaussian games is fixed at 0.5, while the Gaussian variable games use the same distribution of standard deviations as before. BRASS is not included in this set of results because it was too slow and required too much memory to complete for some of the larger problems. The pattern of results is similar to the smaller games. In uniform games there is a greater separation between the algorithms, with ISG outperforming GMC. On Gaussian and Gaussian variable games, GMC has higher solution quality, but the overall difference between GMC and ISG is fairly small.

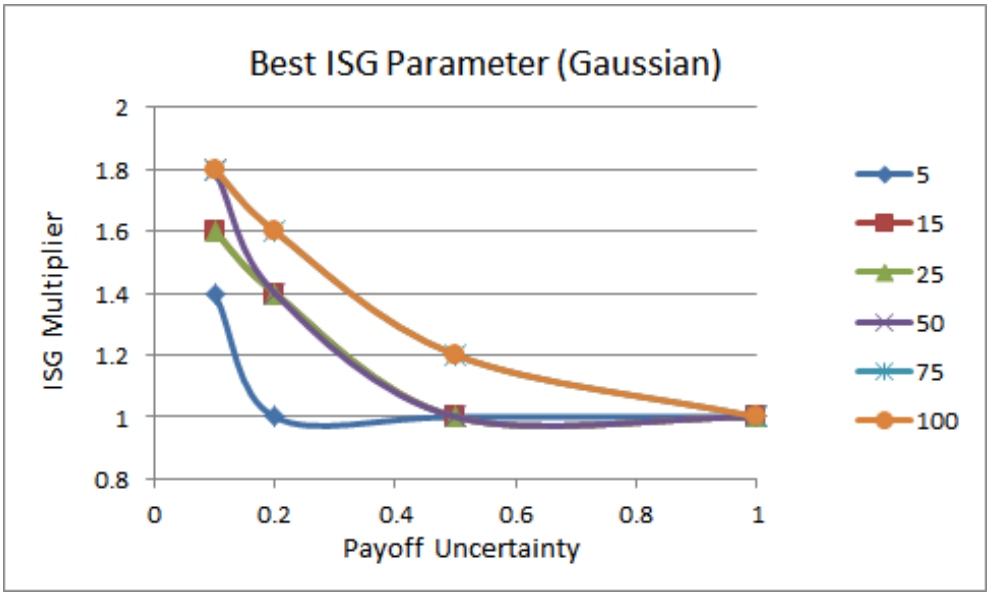

Figure 6.10: Parameter optimization for ISG on games with Gaussian distributions.

Parameters can have an major effect on solution quality and time. There is no obvious way to set the correct value of these parameters. The best value can depend on the size of the game, the type of uncertainty and the amount of uncertainty. For our ISG method the best multiplier value is used (multiplier which gave us maximum defender payoff). Figure 6.10 shows an experiment to find best parameters for games with Gaussian uncertainty for the ISG algorithm. The values of multipliers are placed along the y-axis and the values of payoff uncertainty are placed along the x-axis. Each line in Figure 6.10 represents the

maximum payoff for the given number of targets. In general the best multipliers are smaller for smaller games and games with greater uncertainty. Finding the best parameter for the experiments is not a big problem in practice. The algorithms are fast enough to test a few parameter settings and find the best parameter at any time. The same parameter settings can be used among various solving methods. For BRASS and ISG, all the parameter settings were tested upon and the best one was selected.

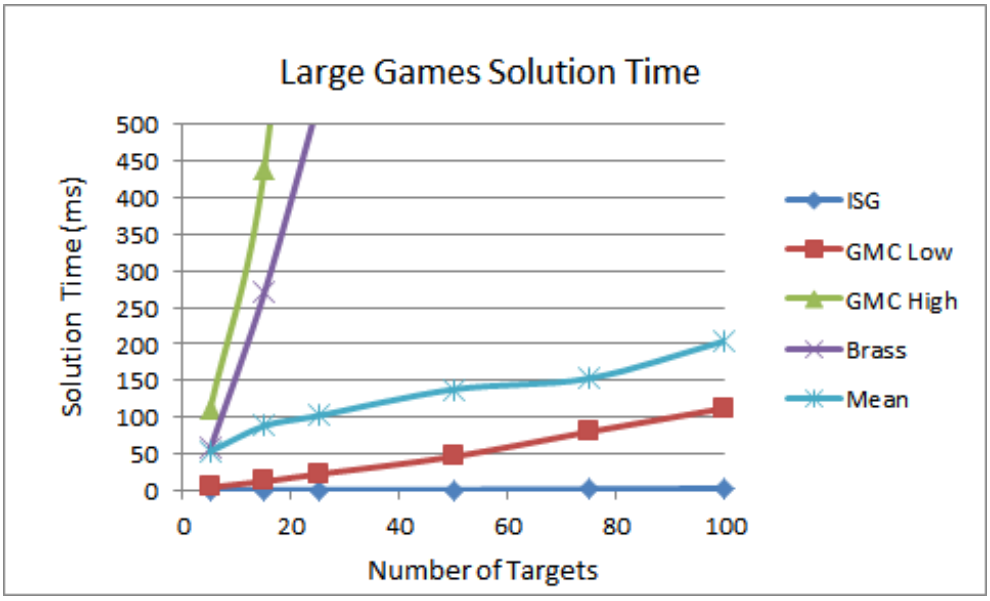

Figure 6.11: Solution quality and computation time comparisons

Figure 6.11 compares all the run-time of various solutions on large Gaussian games. As the size of the game increases, solution times for both BRASS and GMC high rapidly increases. The solution times for Mean and GMC low increases reasonably with game sizes. On the other hand, in this set of solution methods ISG is the fastest algorithm. In a situation where other solution methods take a huge amount time, ISG can provide a solution in minimal amount of time. It provides either competitive or superior solution quality for approximating distributional games, depending on type of uncertainty. For cases where very large games need to be evaluated in short amount of time, ISG is unquestionably the best choice.

6.3 Experiments on Incorrect Game Model

This section describes experiments done to understand the effect of correctness of the game model on solution quality. Defenders may assume a wrong standard deviation for the attacker's payoff function which will lead to a incorrect game model. The defender would like to find solutions that are robust to this kind of incorrect assumptions. For this purpose, the ISG algorithm was executed on the incorrect game model and calculations were done later to find the final defender's payoff based on the correct game model. The steps to calculate defender's payoff based on incorrect game model are shown in the following list.

1. Load correct DSG files
2. Modify the DSG files by changing the standard deviation.
3. Apply ISG algorithm on modified DSG files and get the solution (coverage vector).
4. Save the coverage vector that gives maximum payoff for the defender
5. Load correct DSG files
6. Calculate defender's payoff using best coverage (from step 4) and correct DSG files
7. Take the average over the defender's payoff

Figure 6.12 displays the experimental outcomes for executing ISG algorithm on the incorrect game model. I put the incorrect standard deviation or uncertainty along the x-axis and the defender's payoff along the y-axis. All five lines represent the characteristics for each correct standard deviation. The experiments were done on 100 targets, 20 resources and Gaussian distribution was used to define the attacker's payoff. Figure 6.13 represents the same experiments done on GMC solution method.

Figure 6.12 describes that if the game model is correct (where incorrect and correct standard deviation matches), we have the highest defender's payoff for each of the standard

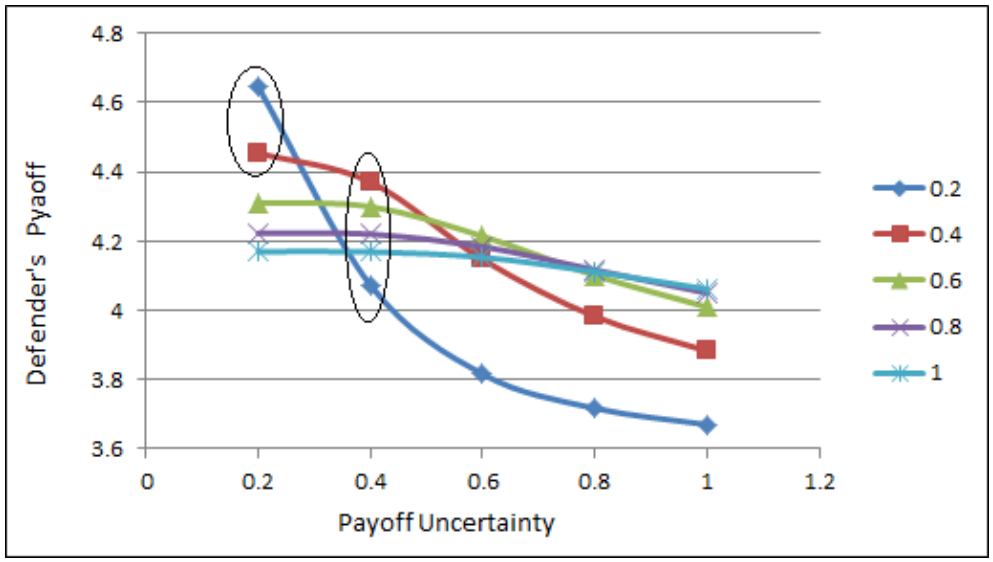

Figure 6.12: Solution quality of ISG on incorrect game model

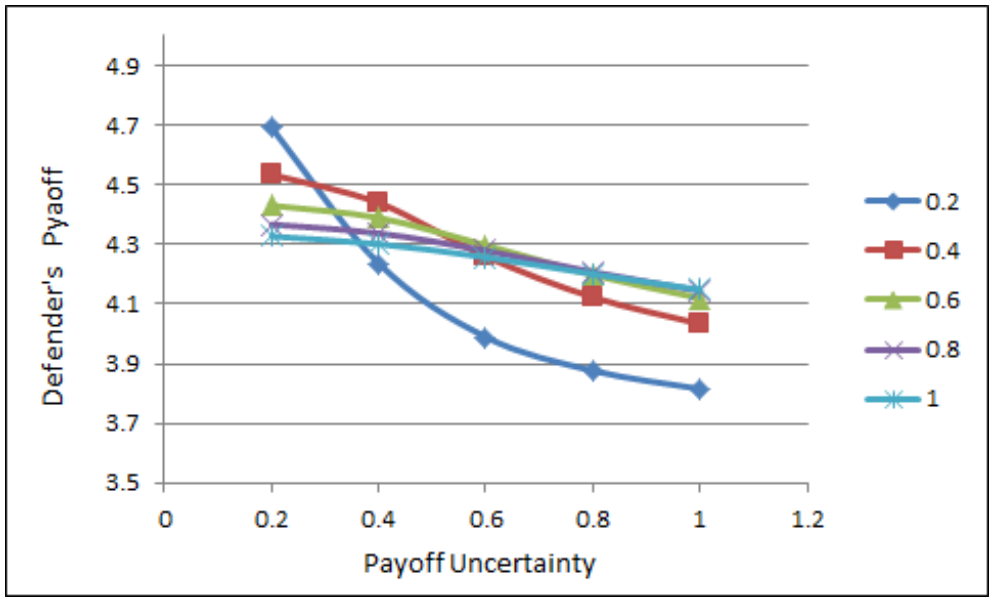

Figure 6.13: Solution quality of GMC on incorrect game model

deviations. However, if the defender assumes an incorrect standard deviation (underestimation or overestimation), he receives a lower payoff value. This same pattern is visible from all five points on top of each of the incorrect standard deviations on the x-axis.

However, another important trait of ISG algorithm is visible from Figure 6.12. Whenever an overestimation is made (incorrect standard deviation is greater than the correct one), the performance of ISG decreases less compared to the case when an underestimation is made (incorrect standard deviation is lower than the correct one). In the circle on the left there is a correct standard deviation of 0.2. But if we assume that standard deviation is 0.4, which is an overestimation, the defender's payoff falls down from 4.6 to 4.4. In the circle to the right, the top square defines the correct standard deviation of 0.4. If the incorrect standard deviation is 0.2, which is an underestimation, the defender's payoff falls from 4.4 to 4.1. So making an underestimation creates more discrepancy in solution quality, where overestimation leads to less erroneous results. I conducted the same set of experiments on GMC with same parameters and found almost similar characteristics.

Chapter 7

Conclusion

Security has become one of the most significant issues for every nation in recent years and applications of security games can play a great role in improving security decisions. However, generating security game models is hindered by the issue of handling uncertainty and errors in building the game models. Also, predicting characteristics of the attackers and estimating their reward values are challenging for domain experts. The solutions for allocating resources must allow for uncertainty while predicting behavior of an adversarial entity, which is a strength of game theoretic modeling. If a proposed solution is not robust in handling uncertainty and errors, it can not be applied to real-world scenarios. Existing solution approaches use Bayesian games to model uncertainty, but those approaches are very challenging from the model generation and computational point of view.

The introduced model handles this uncertainty based on intervals to represent payoffs and takes a worst case approach to uncertainty. This approach is motivated in part by the literature of robust optimization and robust game theory concept. I show that modeling uncertainty using intervals has distinct computational advantages over Bayesian approach. A highly efficient polynomial algorithm is presented for approximating solutions to interval security games. This method has bounded errors and can quickly calculate solutions within very small tolerances of the optimal solution. Empirical results for this algorithm show much faster performance compared to an exact Mixed Integer Programming formulation.

I also showed how distributional uncertainty in an infinite Bayesian Stackelberg game can be converted into interval uncertainty. A methodology was developed to transform the distributional payoff to interval payoff and the fast algorithm, which was developed for interval-based security games, was applied on that game model. To evaluate performance,

the algorithm is tested on three types of distributions: Uniform, Gaussian with fixed standard deviation and Gaussian with variable standard deviation. For Uniform distribution, the ISG method outperforms all other solution methods. In cases of Gaussian and Gaussian variable distributions, the solution quality is very competitive. Based on experimental results, it is concluded that the interval approach is competitive in most cases and for large games it is even better. The speed this novel approach is much faster than all other existing solution methods, so it is concluded that this computationally efficient approach is one of the best ways to solve large security games while still modeling uncertainty about the payoffs.

The final set of experiments was conducted on incorrect game models to show that both underestimation and overestimation of uncertainty on the attacker's payoffs degrade solution quality. However, underestimation of uncertainty will degrade the solution quality more than the case where the uncertainty is overestimated. One of the other major advantages of this interval approach is the simplicity in the generation of game models.

7.1 Future Work

The time and solution quality advantages of the interval based approach may inspire researchers working in the field of security games for future works in adversarial domains. The increased simplicity in generating the game model will allow domain experts to apply this approach in field level applications with revolutionary ideas.

The foundation of this interval security game approach is a Stackelberg game, where the defender (or leader) acts first and the attacker (follower) takes his actions based on observation of defender's strategies. However this assumption can fail if the attacker fails to observe the defender's strategies accurately. If traditional Stackelberg game approach is used, the attacker will end up calculating incorrect payoffs and making a move based on that calculation. In real-life scenarios the defenders may be undercover cops or closed circuit cameras that are hard to observe. If the follower (attacker) observes it wrong, they will

make wrong moves and convert the Stackelberg game model into a simultaneous-move game model. Previously deployed security game applications did not handle this fundamental uncertainty but there are some ongoing work on this issue [17]. So handling observational uncertainty using intervals will be a notable extension of this work. The interval based approach can also be used for more general classes of games (i.e., normal form games) with simultaneous moves, and without the security game payoff structures.

References

[1] M. Aghassi and D. Bertsimas. Robust game theory. *Mathematical Programming*, Volume 107, Pages 231–273, 2006.

[2] N. Agmon, S. Kraus, G. Kaminka, and V. Sadov. Adversarial uncertainty in multi-robot patrol. In *Proceedings of the 21st International Joint Conference on Artifical intelligence IJCAI*, Pages 1811–1817, Pasadena, California, July 11–17, 2009.

[3] T. Alpcan and T. Basar. A game theoretic approach to decision and analysis in network intrusion detection. In *Proceedings of the 42nd IEEE Conference on Decision and Control*, Pages 2595–2600, Maui, Hawaii, Dec 9–12, 2003.

[4] N. Basiloco, N. Gatti, and F. Amigoni. Leader-follower strategies for robotic patrolling in environments with arbitrary topologies. In *Proceedings of 8th International Conference on Autonomous Agents and Multiagent Systems AAMAS'09*, Pages 57–64, Budapest, Hungary, May 10–15, 2009.

[5] V. Bier. Choosing what to protect. *Risk Analysis*, Volume 27, Issue 3, Pages 607–620, 2007.

[6] G. Cai and P. Wurman. Monte Carlo approximation in incomplete information, sequential auction games. *Decision Support Systems*, Volume 39, Issue 2, Pages 153–168, 2005.

[7] T. Carroll and D. Grosu. A game theoretic investigation of deception in network security. In *Proceedings of 18th International Conference on Computer Communications and Networks*, Pages 1–6, Nassau, Bahamas, July 30 – August 2, 2013.

[8] V. Conitzer and T. Sandholm. Computing the optimal strategy to commit to. In *Proceedings of the 7th ACM Conference on Electronic Commerce*, Pages 82–90, Ann Arbor, Michigan, June 11–15, 2006.

[9] N. Gatti. Game theoretical insights in strategic patrolling: Model and algorithm in normal-form. In *Proceedings of ECAI 2008: 18th European Conference on Artificial Intelligence*, Pages 403–407, Patras, Greece, July 21–25, 2008.

[10] GLPK. http://www.gnu.org/software/glpk/, 2013.

[11] J. Harsanyi. Games with incomplete information played by Bayesian players (parts i–iii). *Management Science*, Volume 14, Pages 159–182, 1967.

[12] M. Jain, M. Tambe, and C. Kiekintveld. Quality-bounded solutions for finite Bayesian Stackelberg games: scaling up. In *Proceedings of the 10th International Conference on Autonomous Agents and Multiagent Systems AAMAS'11*, Pages 997–1004, Taipei, Taiwan, May 2–6, 2011.

[13] L. Jaulin, K. Michel, D. Olivier, and E. Walter. Applied Interval Analysis. *Springer, Berlin*, 2001.

[14] R. Kearfott and V. Kreinovich. Applications of Interval Computations. *Kluwer Academic Publishers, Dordrecht*, 1996.

[15] C. Kiekintveld, M. Jain, J. Tsai, J. Pita, F. Ordonez, and M. Tambe. Computing optimal randomized resource allocations for massive security games. In *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems AAMAS'09*, Volume 1, Pages 689–696, Budapest, Hungary, May 10–15, 2009.

[16] C. Kiekintveld, J. Marecki, and M. Tambe. Approximation methods for infinite Bayesian Stackelberg games: Modeling distributional payoff uncertainty. In *Proceedings of The 10th International Conference on Autonomous Agents and Multiagent Systems AAMAS'11*, Volume 3, Pages 1005–1012, Taipei, Taiwan, May 2–6, 2011.

[17] D. Korzhyk, Z. Yin, C. Kiekintveld, V. Conitzer, and M. Tambe. Stackelberg vs. Nash in security games: An extended investigation of interchangeability, equivalence, and uniqueness. *Journal of Artificial Intelligence Research*, Volume 41, Issue 2, Pages 297–327, 2011.

[18] V. Krenovich. Applications of interval computation. http://www.cs.utep.edu/interval-comp/main.html, 2013.

[19] V. Krishna. *Auction Theory*. Academic Press, Burlington, 2002.

[20] R. Luce and H. Raiffa. *Games and Decisions*. John Wiley and Sons, New York, 1957. Dover republication, 1989.

[21] D. McFadden. Quantal choice analysis: A survey. *Annals of Economic and Social Measurement*, Volume 5, Pages 363–390, 1976.

[22] K. Nguyen, T. Alpcan, and T. Basar. Security games with incomplete information. In *Proceedings of the 2009 IEEE international Conference on Communications*, Pages 714–719, Dresden, Germany, June 14–18, 2009.

[23] P. Paruchuri, J. Pearce, J. Marecki, M. Tambe, F. Ordonez, and S. Kraus. Playing games with security: An efficient exact algorithm for Bayesian Stackelberg games. In *Proceedings of the 8th International Conference on Autonomous Agents and Multiagent Systems AAMAS'09*, Pages 895–902, Budapest, Hungary, May 10–15, 2009.

[24] J. Pita, M. Jain, F. Ordóñez, M. Tambe, S. Kraus, and R. Magori-Cohen. Effective solutions for real-world Stackelberg games: When agents must deal with human uncertainties. In *Proceedings of the 8th International Conference on Autonomous Agents and Multiagent Systems AAMAS'09*, Pages 369–376, Budapest, Hungary, May 10–15, 2009.

[25] J. Pita, M. Jain, C. Western, C. Portway, M. Tambe, F. Ordonez, S. Kraus, and P. Parachuri. Deployed ARMOR protection: The application of a game-theoretic

model for security at the Los Angeles International Airport. In *Proceedings of the 7th International Conference on Autonomous Agents and Multiagent Systems AAMAS'08*, Estoril, Portugal, May 12–16, 2008.

[26] J. Pita, M. Tambe, C. Kiekintveld, S. Cullen, and E. Steigerwald. GUARDS – game theoretic security allocation on a national scale. In *Proceedings of the 7th International Conference on Autonomous Agents and Multiagent Systems AAMAS'11*, Pages 37–44, Taipei, Taiwan, May 2–6, 2011.

[27] T. Sandler and D. G. Arce M. Terrorism and game theory. *Simulation and Gaming*, Volume 34, Issue 3, Pages 319–337, 2003.

[28] E. Shieh, B. An, R. Yang, M. Tambe, C. Baldwin, J. Direnzo, G. Meyer, C. W. Baldwin, and B. J. Maule. PROTECT: A deployed game theoretic system to protect the ports of the United States. *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems AAMAS'12*, Volume 1, Pages 13–20, Valencia, Spain, June 4–8, 2012.

[29] J. Tsai, S. Rathi, C. Kiekintveld, F. Ordóñez, and M. Tambe. IRIS – A tool for strategic security allocation in transportation networks. In *Proceedings of the 8th International Conference on Autonomous Agents and Multiagent Systems AAMAS'09*, Pages 369–376, Budapest, Hungary, May 10–15, 2009.

[30] K. Lye and J. Wing. Game strategies in network security. *International Journal of Information Security*, Volume 4, Issue 1–2, Pages 71–86, 2005.

[31] Z. Yin and M. Tambe. A unified method for handling discrete and continuous uncertainty in Bayesian Stackelberg games. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems AAMAS'12*, Volume 2, Pages 855–862, Valencia, Spain, June 4–8, 2012.

Curriculum Vitae

Md Towhidul Islam was born in Dhaka, Bangladesh on December 12, 1987. He graduated from Bangladesh University of Engineering and Technology (BUET) with a bachelor's in Computer Science and Engineering degree in 2009. After completing his undergrad study, he joined one of the largest telecommunication companies in Bangladesh, Robi Axiata Ltd. He worked there for two years in Data Warehouse Development team as a specialist.

In the fall of 2011, he entered the Graduate School of The University of Texas at El Paso. While pursuing a master's degree in Computer Science he worked as a Teaching Assistant for the course of Elementary Data Structure and Algorithm. He also worked as a Research Assistant in The Intelligent Agents and Strategic Reasoning Lab (IASRL).

Permanent address: 716, West Yandell Dr, Apt 17
El Paso, Texas 79902