Opportunistic Crime Security Games: Assisting Police to Control Urban Crime Using Real World Data

by

Chao Zhang

A Dissertation Presented to the
FACULTY OF THE GRADUATE SCHOOL
UNIVERSITY OF SOUTHERN CALIFORNIA
In Partial Fulfillment of the
Requirements for the Degree
DOCTOR OF PHILOSOPHY
(COMPUTER SCIENCE)

April 2016

Acknowledgements

First, I would like to give my special thanks to my advisor, Professor Milind Tambe, for his guidance throughout my PhD career. Milind, I thank you for discovering my capacity and bringing me in Teamcore group. It is this precious opportunity that led me in the world of research. I thank you for your patience in me. When I first joined Teamcore group, I did not know much about research and I had to learn from the beginning. You are so patient and gave me detailed instructions step by step. I thank you for your dedication for me, even though you are such a busy person, you regularly meet with me to discuss my research progress and always there to answer my questions and clean my doubts in research. I really appreciate the time and effort you spent in me. I also thank you for your attitude in research. You are always passionate about research. I am strongly impressed by your attitude. When my research stagnates and I am about to give up on that direction, it is your optimistic attitude that makes me to try one more time and succeed. In addition, you are not only a great advisor for research, but also an excellent tutor in life. I thank you for all your advice you gave me on my life path. I thank you for helping me plan my career and sharing your experience with me.

Second, I would like to thank my committee members for their insights and feedback in my research: Cyrus Shahabi, Kevin Knight and Najmedin Meshkati. Cyrus, thank you

very much for inviting me to your group for talking about my research. You suggestions and comments help me a lot in my research. Kevin, that you very much for discussing with me about my research. Your insights and experience in machine learning have a great contribution to the learning model in my game. Najmedin, thank you so much for agreeing to be my committee. Your insights from outside CS give me a new point of view for my research.

Next, I would like to thank the excellent colleagues that I have had the privilege to work with during my PhD. This list includes Albert Xin Jiang, Zhengyu Yin, Sarit Kraus, Francesco Maria Delle Fave, Eric Shieh, Martin Short, Jeffrey Brantingham, Leandro Soriano Marcolino, Arunesh Sinha, Yasaman Dehghani Abbasi, Shahrzad Gholami, Victor Bucarey, Ayan Mukhopadhyay, Yevgeniy Vorobeychik and Yundi Qian. Albert, I thank you for helping me build the original model. Zhengyu, I thank you for letting me working with your project. Sarit, I thank you for working with me and giving me advice for job searching. Francesco and Eric, I thank you for doing field experiments for our model. Martin and Jeff, I thank you for the knowledge your bring from math and criminology domain. Leandro, I thank you for playing go and doing research on go together. Arunesh, I thank you for all your help in my research in my third and fourth year. Yasi, I thank you for running human experiments for our model. Shahrzad, I thank you for building up the variation of our model. Victor, I thank you for helping me process the geographic data. Ayan and Eugene, I thank you for running our model in your valuable datasets. Yundi, I thank you for your advice and insights in my model.

Also, I would like to thank all the members of TEAMCORE family: Albert Jiang, Francesco Delle Fave, William Haskell, Arunesh Sinha, Zhengyu Yin, James Pita, Jason

Tsai, Manish Jain, Jun-young Kwak, Rong Yang, Eric Shieh, Matthew Brown, Fei Fang, Thanh Hong Nguyen, Leandro Marcolino, Yundi Qian, Debarun Kar, Benjamin Ford, Haifeng Xu, Kumar Amulya Yadav, Aaron Schlenker, Sara Marie Mc. Carthy, Yasaman Dehghani Abbasi, Shahrzad Gholami, Byran Wilder and Elizabeth Orrico. It is my great honor to share my PhD experience with all of you and I really enjoy the time that we work and play together.

Finally I would like to thank my friends and families, who gave me tremendous help for me to complete my PhD. I would like to thank my parents, Huping Zhang and Hanping Cao, who support me both economically and mentally. They call me weekly to relief my pressure from work and study and always be there when I need their help. Their support is the preliminary condition for me to complete my PhD. I would like to thank my girlfriend Qianwen Li for her support and my roommate Shuang Chen for listening to me and discussing with me about my research.

Table of Contents

List Of Figures

Abstract

Crime in urban areas plagues every city in all countries. A notable characteristic of urban crime, distinct from organized terrorist attacks, is that most urban crimes are opportunistic in nature, i.e., criminals do not plan their attacks in detail, rather they seek opportunities for committing crime and are agile in their execution of the crime. In order to deter such crimes, police officers conduct patrols with the aim of preventing crime. However, by observing on the spot the actual presence of patrol units, the criminals can adapt their strategy by seeking crime opportunity in less effectively patrolled location. The problem of where and how much to patrol is therefore important.

My thesis focuses on addressing such opportunistic crime by introducing a new game-theoretic framework and algorithms. I first introduce the Opportunistic Security Game (OSG), a computational framework to recommend deployment strategies for defenders to control opportunistic crimes. I propose a new exact algorithm EOSG to optimize defender strategies given our opportunistic adversaries. Then I develop a fast heuristic algorithm to solve large-scale OSG problems, exploiting a compact representation. The next contribution in my thesis is a Dynamic Bayesian Network (DBN) to learn the OSG model from real-world criminal activity. Standard Algorithm such as EM can be applied to learn the parameters. Also, I propose a sequence of modifications that allows for a

compact representation of the model resulting in better learning accuracy and increased speed of learning of the EM algorithm. Finally, I propose a game abstraction framework that can handle opportunistic crimes in large-scale urban areas. I propose a planning algorithm that recommends a mixed strategy against opportunistic criminals in this abstraction framework. As part of our collaboration with local police departments, we apply our model in two large scale urban problems: USC campus and the city of Nashville. Our approach provides high prediction accuracy in the real datasets; furthermore, we project significant crime rate reduction using our planning strategy compared to current police strategy.

Chapter 1

Introduction

Security is a critical societal challenge. We focus on urban security: the problem of preventing urban crimes. Crime in urban areas plagues every city in all countries. A notable characteristic of urban crime, distinct from organized terrorist attacks, is that most urban crimes are opportunistic in nature, i.e., criminals do not plan their attacks in detail, rather they seek opportunities for committing crime and are agile in their execution of the crime.

The Stackelberg Security Game (SSG) was proposed to model highly strategic and capable adversaries who conduct careful surveillance and plan attacks [32, 53], and has become an important computational framework for allocating security resources against such adversaries. While there are such highly capable adversaries in the urban security domain, they likely comprise only a small portion of the overall set of adversaries. Instead, the majority of adversaries in urban security are criminals who conduct little planning or surveillance before "attacking" [13]. These adversaries capitalize on local opportunities and react to real-time information. Unfortunately, SSG is ill-suited to model such criminals, as it attributes significant planning and little execution flexibility to adversaries.

1.1 Problem Addressed

In my thesis I address three questions. First, managing urban crime has always posed a significant challenge for modern society. Distinct from elaborately planned terrorists attacks, urban crimes are usually committed by *opportunistic criminals* who are less careful in planning the attack and more flexible in executing such plans [51]. Almost universally, preventive police patrolling is used with the goal of deterring these crimes. At the same time, opportunistic criminals observe the police deployment and react opportunistically. Therefore, it is very important to deploy the police resources strategically against informed criminals.

The first approach has focused on modeling the criminal explicitly (rational, bounded rational, limited surveillance, etc.) in a game model. However, the proposed mathematical models of criminal behavior have not been validated with real data. Therefore, we tackle the problem of learning the opportunistic criminal behavior from real data.

Thirdly, allocating police resources against opportunistic criminals in large scale urban areas remains to be an open question. While security games contain various extensions to handle different real world scenarios, the models of adversary behavior are based on expert hypotheses, and lack detail as they are not learned from real-world data for defender's strategy and adversary's reaction. While the second approach uses larger amounts of data, such as the patrol allocation history and corresponding crime report, to learn a richer Dynamic Bayesian Network (DBN) model of the interaction between the police officers and opportunistic criminals. The optimal patrol strategy is generated using the learned parameters of the DBN. While this approach predicts criminals' behavior with

high accuracy for the problem in which the number of target areas is small, it has three shortcomings: i) it cannot scale up to problems with a large number of targets; ii) the algorithm performs poorly in situations where the defender's patrol data is limited; iii) the planning algorithm only searches for a pure patrol strategy, which quickly converges to a predictable pattern that can be easily exploited by criminals.

1.2 Contributions

My thesis addresses the questions raised in previous section to control the opportunistic criminals in urban areas.

1.2.1 Opportunistic Security Games

First, I introduces the Opportunistic Security Game (OSG), a new computational framework for generating defender strategies to mitigate opportunistic criminals. This model provides three key contributions. First, we define the OSG model of opportunistic criminals, which has three major novelties compared to SSG adversaries: (i) criminals exhibit Quantal Biased Random Movement, a stochastic pattern of movement to search for crime opportunities that contrasts with SSG adversaries, who are modeled as committed to a single fixed plan or target; (ii) criminals react to real-time information about defenders, flexibly altering plans during execution, a behavior that is supported by findings in criminology literature [46]; (iii) criminals display anchoring bias [49], modeling their limited surveillance of the defender's strategy. Second, we introduce a new exact algorithm, Exact Opportunistic Security Game (EOSG), to optimize the defender's strategy in OSG based on use of a markov chain. The third contribution of this work is a fast algorithm,

Compact OPportunistic Security game states (COPS), to solve large scale OSG problems. The number of states in the Markov chain for the OSG grows exponentially with the number of potential targets in the system, as well as with the number of defender resources. COPS compactly represents such states, dramatically reducing computation time with small sacrifice in solution quality; we provided a bound for this error.

Thus, while OSG does share one similarity with SSG — the defender must commit to her strategy first, after which the criminals will choose crime targets — the OSG model of opportunistic adversaries is fundamentally different. This leads us to derive completely new algorithms for OSG. OSG also differs fundamentally from another important class of games, pursuit-evasion games (PEG) [26]; these differences will be discussed in more depth in the related work section.

While OSG is a general framework for handling opportunistic crime, my thesis will use as a concrete example crime in urban transportation systems, an important challenge across the world. Transportation systems are at a unique risk of crime because they concentrate large numbers of people in time and space [14]. The challenge in controlling crime can be modeled as an OSG: police conduct patrols within the transportation system to control crime. Criminals travel within the transportation system for such opportunities [19], usually committing crimes such as thefts at stations, where it is easy to escape if necessary [35]. These opportunistic criminals avoid committing crime if they observe police presence at the crime location.

In introducing OSG, my thesis proposes to add to the class of important security related game-theoretic frameworks that are widely studied in the literature, including the Stackelberg Security Games and Pursuit Evasion Games frameworks. We use an

urban transportation system as an important concrete domain, but OSG's focus is on opportunistic crime in general; the security problems posed by such crime are relevant not only to urban crime, but to other domains including crimes against the environment [45], and potentially to cyber crime [17,61]. By introducing a new model and new algorithms for this model, we open the door to a new set of research challenges.

1.2.2 Learning OSG from real data

Next, we tackle the problem of learning the criminal behavior from real data. We do so by modeling the interaction between the criminal and patrol officers as a Dynamic Bayesian Network (DBN). This DBN model is our *first contribution*. As far as we know, we are the first to use a DBN model that considers the temporal interaction between defender and adversary in the learning phase. Given a DBN model, we can use the well-known Expectation Maximization (EM) algorithm to learn unknown parameters in the DBN from given learning data. However, using EM with the basic DBN model has two drawbacks: (1) the number of unknown parameters scales exponentially with the number of patrol areas and in our case is much larger than the available data itself; this results in over-fitting (2) EM cannot scale up due to the exponential growth of runtime in the number of patrol areas. We demonstrate these two drawbacks both theoretically and empirically.

Our *second contribution* is a sequence of modifications of the initial DBN model resulting in a compact representation of the model, that leads to better learning accuracy and increased speed of learning of the EM algorithm when used for the compact model.

This sequence of modifications involve marginalizing states in the DBN using approximation technique from the Boyen-Koller algorithm [12] and exploiting structure of this problem. In the compact model, the parameters scale polynomially with the number of patrol areas, and EM applied to this compact model runs in polynomial time.

Our *third contribution* are two planning algorithms that enable computing the optimal officers' strategy. First, we present a dynamic programming based algorithm that computes the optimal plan in our planning and updating process. While the dynamic programming approach is optimal, it may be slow, hence we also present a fast but sub-optimal greedy algorithm to solve the planning problem. Further, the criminal behavior would change as he observes and reacts to the deployment of a new strategy. Hence, the optimal strategy with respect to the learnt behavior may not be effective for a long time, as the adversary behavior would have changed. Thus, we propose to frequently update our adversary model as we obtain new training data from a new deployment of defender strategy. By repeating the planning and updating process, we recommend officers' strategy that are more effective than learning just once.

Finally, as part of our collaboration with the police department of USC, we obtained criminal activity and patrol data for three years. This collaboration helped us validate our learning approach and also provided insights about the sequence of modifications that could be made for the basic DBN model. In fact, we project a significant reduction in crime rate using our approach as opposed to the current patrolling approach (see Figure 4.20). Given these results, we expect our algorithm to be tested and eventually deployed in USC. More broadly, by introducing a novel framework to reason about urban

crimes along with efficient learning and planning algorithms, we open the door to a new set of research challenges.

1.2.3 Learning large scale OSGs

Finally, we focus on the problem of generating effective patrol strategies against opportunistic criminals in large scale urban settings. In order to utilize the superior performance of DBN as compared to other models given ample data, we propose a novel abstraction framework. This abstraction framework is our *first contribution*. In this framework we merge the targets with similar properties and extract a problem with a small number of targets. We call this new problem the *abstract layer* and the original problem the *original layer*. We first learn in the abstract layer using the DBN approach [60] and generate the optimal patrol strategy, then we propagate the learned parameters to the original layer and use the resource allocation in the abstract layer to generate a detailed strategy in the original layer. By solving the problem hierarchically through multiple abstractions, we can generate the optimal strategy for the original scenario.

Our *second contribution* is a layer generating algorithm, for which (i) we model it as a districting problem and propose a MILP in order to merge targets in the original problem into geographically compact and contiguous *aggregated targets* keeping the similarity (defined later) within them as homogeneous as possible; (ii) we develop a heuristic to solve this problem in large scale instances; (iii) we propose two approaches to find the optimal aggregated targets. Our *third contribution* is a planning algorithm that generates an optimal mixed strategy against opportunistic criminals. We consider a mixed strategy because (i) it broadens the scope of the defender's strategies; (ii) previous pure strategies

depended on the model getting updated periodically; as mentioned earlier, the model usually converged to a single pure strategy that is easy to exploit.

When the defender's patrol data is limited or even missing in the original layer, the learning approach in [60] overfits the data. Therefore, in order to solve this problem, we propose our *fourth contribution* which is a heuristic model to propagate important features from the abstract layer to the original layer. We use models from behavioral game theory, such as Quantal Response, to extract these features. In particular, we first approximate the learned DBN parameters in the abstract layer using behavioral parameters. Then the behavioral parameters are propagated to the original layer.

Finally, we evaluate our abstract game in two scenarios: the University of Southern California (USC) campus [60] and Nashville, TN. We obtain data in USC from [60]. Data in Nashville, TN is obtained as part of the collaboration with the local police department.

1.3 Overview of Thesis

This thesis is organized as follows. Chapter 2 introduces the necessary background materials for the research presented in this thesis. Chapter 3 provides an overview of the related work. Chapter 4 discusses the Opportunistic Security Game framework and Compact OPportunistic Security game states algorithm. Chapter 5 presents the framework of learning criminal's behavior in OSG from real world data. Chapter 6 provides an extension of the existing learning framework to further improve its performance in large scale problems. Finally, Chapter 7 concludes the thesis and presents possible future directions.

Chapter 2

Background and Related Work

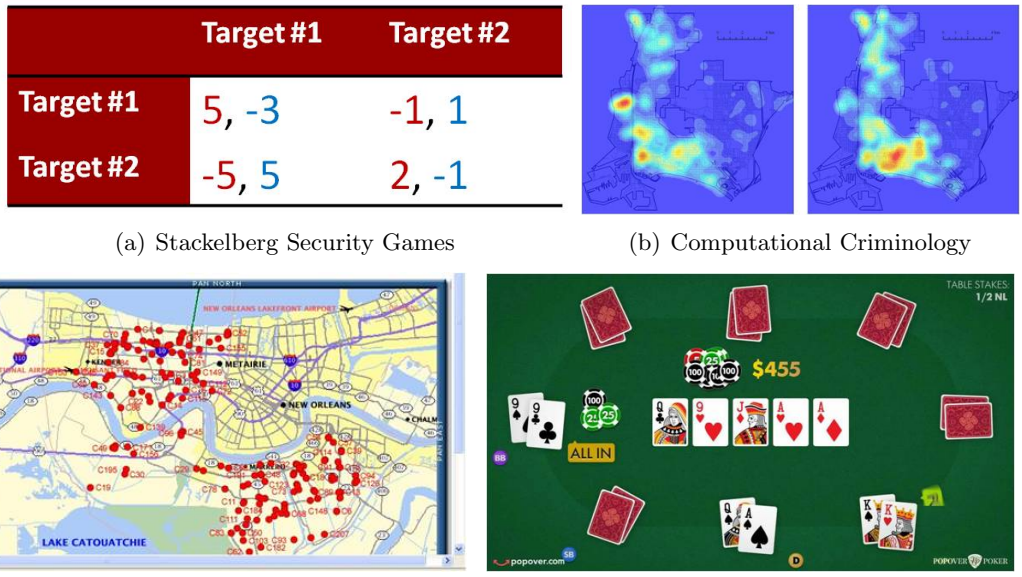
Motivated by urban security problems, there have been a lot research for learning criminal's behavior computing optimal defender strategies. We categorize the related work into five main areas. The first area is game theoretic models such as Stackelberg Security Games (SSG) and Pursuit Evasion Games (PEG). The second thread of recent research has made inroads in the modeling of opportunistic criminal behavior. The third kind of research applies machine learning technique in criminology to predict crimes. The fourth thread of research uses machine learning technique in game theory. The last thread of research we comapred with is the abstract game.

2.1 Game Theoretic Models

Game theoretic models, such as Stackelberg Security Games (SSG) [53, 55], Patrolling security games (PSG) [6] and Pursuit Evasion Games (PEG) [26] have been widely used in security domains. The interaction between police and criminals is modeled as a Security Game. Stackelberg Security Games [3, 5, 7, 55], use a model of highly strategic adversaries to generate randomized patrol strategies, which is shown in Fig. 2.1(a). The

SSG framework has been successfully applied in security domains to generate randomized patrol strategies, e.g., to protect flights [53] and for counter-terrorism and fare evasion checks on trains [28, 58]. Recent work in SSG has begun to consider bounded rationality of adversaries [43] and incorporate some limited flexibility in adversary execution [7, 57]. However, SSG [3, 5, 7] fails to model criminals who use real-time information to adjust their behavior in consecutive multiple attacks. In SSG, attackers cannot use real-time observation to decide whether to attack at current time, nor can they use it to update belief and plan for next consecutive attacks. Furthermore, SSG does not investigate efficient algorithms of deriving defender strategies against such opportunistic criminals; Adversarial Patrolling Game(APG) [56], which is an improvement of SSG, considers the attacker's current observation. However, this game does not consider multiple consecutive attacks. It fails to model attacker's movement during multiple attacks and therefore the influence of current observation on future movement.

Recent work in leader-follower games, PSG, also has made progress in generating patrol strategies against adversaries in arbitrary topology [5]. Different types of adversaries in this game are considered in [7] while different security resources are considered in [4]. Another example of a game theoretic model is Pursuit-Evasion Games (PEG), which model a pursuer(s) attempting to capture an evader, often where their movement is based on a graph [26]. However, PEG fail to model criminals who opportunistically and repeatedly strike targets as modeled using QBRM in OSG. Furthermore, in PEG, a pursuer's goal is to capture an evader while in OSG, the defender's goal is to minimize crime; additionally in PEG, the evader's goal is to avoid the pursuer and not seek crime



(a) Stackelberg Security Games

(b) Computational Criminology

(c) Machine Learning in Criminology

(d) Texas Hold'em

Figure 2.1: Related Work

opportunities as in OSG. These critical differences in behaviors of defenders and adversaries lead to new algorithms, i.e., EOGS and COPS, for OSG, that are fundamentally different from algorithms for PEG.

2.2 Modeling criminal behavior

Computational criminology is an interdisciplinary field that uses computing science and math methods to formally define criminology concepts, improve our understanding of crime distributions, and generate optimal patrol strategies against such crimes. Recent research has also made inroads in the modeling of opportunistic criminal behavior, and in how security forces might defend against such adversaries. In [51], which is shown in Fig. 2.1(b), burglars are modeled as biased random walkers seeking "attractive" targets, and [62] follows up on this work with a method for generating effective police allocations to combat such criminals. However, these works make the extreme assumption

that criminals have no knowledge of the overall strategy of the police, and their behavior is only affected by their observation of the current police allocation in their immediate neighborhood. Also, in [62] police behave in a similarly reactionary way, allocating their resources in an instantaneously optimal way in response to the current crime risk distribution rather than optimizing over an extended time horizon. Furthermore, in [62] there is no notion of the "movement" of police - rather, the distribution of police officers are chosen instantaneously, with no regard for the mechanics of exactly how the allocation may transform from one time step to the next. Our current approach is an attempt to generalize these threads of research. A number of sophisticated modeling approaches emerged aiming to tackle the full spatio-temporal complexity of crime dynamics. One of these is based on a spatio-temporal differential equation model that captures both spatial and temporal crime correlation [38]. These models have two disadvantages: first, they do not naturally capture crime co-variates, and second, they are non-trivial to learn from data. Another model in this general paradigm is Dynamic Spatial Disaggregation Approach (DSDA) [27], which combines an autoregressive model to capture temporal crime patterns with spatial clustering techniques to model spatial correlations. An alternative approach, risk-terraine modeling, focuses on quantifiable environmental factors as determinants of spatial crime incidence, rather than looking at crime correlation [51]. These two classes of models both have a key limitation: they ignore the temporal dynamics of crime. Moreover, environmental risk factors and spatial crime analysis are likely complementary. My approach aims to merge these ideas in a principled way.

2.3 Machine Learning in Criminology

Machine learning is a subfield of computer science, which extracts patterns and learns parameters. In our case, we are specifically referring to supervised machine learning, where the computer is presented with data that has labels and features. The goal of supervised learning is to find a pattern that maps features to labels. More specifically, we are looking at problems that does not only includes features, but also depends on unobserved latent variables. Recent research on Expectation Maximization algorithm has provided an iterative method for finding maximum likelihood of parameters in such statistical models.

Recent research applies machine learning and data mining in criminology domain to analyze crime patterns and support police in making decisions. A general framework for crime data mining is introduced in [16]. In [40], data mining is used to model crime detection problems and cluster crime patterns; in [20], data mining approaches are applied in criminal career analysis; in [42], the authors apply machine learning techniques to soft forensic evidence and build decision support systems for police. However, this area of research considers only crime data and does not model the interaction between patrol officers and criminals. Such works are shown in Fig. 2.1(c)

There has also been an extensive literature devoted to understanding and predicting crime incidence, involving both qualitative and quantitative approaches. For example, a number of studies investigate the relationship between liquor outlets and crime [52, 54]. Many of the earlier quantitative models of crime focus on capturing spatial crime correlation (hot spots), and make use of a number of statistical methods towards this end [34,39];

these are still the most commonly used methods in practice. An alternative approach, risk-terraine modeling, focuses on quantifiable environmental factors as determinants of spatial crime incidence, rather than looking at crime correlation [31]. These two classes of models both have a key limitation: they ignore the temporal dynamics of crime. Moreover, environmental risk factors and spatial crime analysis are likely complementary. Our approach aims to merge these ideas in a principled way.

2.4 Machine Learning in Game Theory

Recent research combines machine learning with game theory. In [10], the defender's optimal strategy is generated in a SSG by learning the payoffs of potential attackers from their best responses to defender's deployments. An inherent problem with such an approach is that the defender strategy is geared towards learning the adversary payoff, and not exploiting the improved knowledge of the adversary payoff as the game progresses. Another example of such work is Green Security Games (GSG) [22, 41], where poaching data may be used to learn a model of poachers' boundedly rational decision making.

2.5 Abstract Game

Abstract game that is widely used in large incomplete information games such as Texas Hold'em that is shown in Fig. 2.1(d), [23, 25]. There are a number of different approaches including both lossless abstractions [24] and lossy abstractions [48]. In [18] and [2], sub-games are generated to calculate the Nash equilibrium in a normal form games. Abstractions have also been brought into security games. In [3], abstraction is

used to design scalable algorithms in PSGs. However, these works focus on clustering similar actions, strategies or states to formulate a simpler game. In our situation, we are physically merging the similar targets to generate simpler games. The criteria of merging targets is different from that of merging actions, strategies or states. Our differing criteria and approach for merging targets, different means of propagating results of our abstractions, and our learning from real-world crime data set our work apart from this work.

Chapter 3

Opportunistic Security Games

In this chapter, I introduce the Opportunistic Security Game (OSG), a computational framework to recommend deployment strategies for defenders to control opportunistic crimes.

3.1 OSG Framework

OSG unfolds on a connected graph that can be seen to model a metro rail system (though many other domains are also possible), where stations are nodes and trains connecting two stations are edges. Fig. 3.1 shows a simple scenario with three fully connected stations. Stations and trains are collectively referred to as locations. Let the stations be labeled $1, \ldots, N$, with N denoting the number of stations. The train from station i to its neighboring station j is denoted as $i \to j$. The number of locations is $N_l > N$, e.g., in Fig. 3.1, $N_l = 9$.

We divide time equally into time steps so that trains arrive at stations at the beginning of each time step. There are two phases in any time step. First is the *decision phase*, the period when trains are at stations for boarding and unboarding. In this phase, each

passenger at each location decides where in the system to move next. There are two types of choices available. *Go* $i \to j$ means that (i) if a passenger is at station i, he gets on the train $i \to j$; (ii) if he is on a train arriving at station i, he now gets (or stays) on the train $i \to j$. *Stay* means that the passenger stays at the station, so that if the passenger was on a train, he gets off. After the brief decision phase is the *action phase*, in which trains depart from all stations to all directly connected stations. This model matches the metro systems in Los Angeles, where trains leave stations at regular intervals to all directly connected stations. Without losing generality, we assume that the time it takes to travel between any two adjacent stations is identical; this assumption can be relaxed by including dummy stations. In OSG, the defender ("she") – assisted by our algorithms – is modeled to be perfectly rational. The criminal ("he") is modeled with cognitive biases. Fig. 3.2 illustrates the OSG flowchart, with relevant equation numbers near variables – these variables and equations are described in the following.

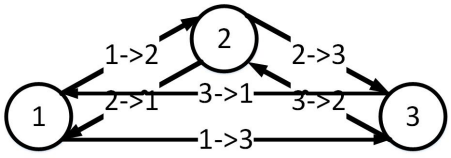


Figure 3.1: The metro network

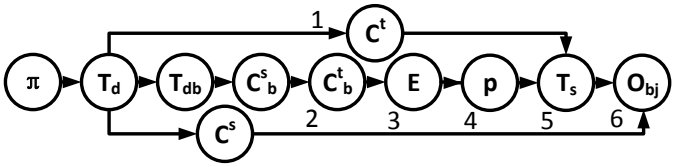


Figure 3.2: Flow chart of OSG

3.1.1 Modeling Defenders

A defender is a team of police officers using trains for patrolling to mitigate crime. We start with a single defender and deal with multiple defenders later. The defender conducts randomized patrols using a *Markov Strategy* π, which specifies for each location a

probability distribution over all available actions. At location l, the probabilities of *Go* $i \to j$ and *Stay* are denoted by $g_l^{i\to j}$ and s_l respectively.

Example 1: Markov Strategy *In Figure 3.1, a possible distribution for location* $3 \to 2$ *in a Markov strategy* π *is,*

$$s_{3\to 2} = 0.1,\ g_{3\to 2}^{2\to 1} = 0.8,\ g_{3\to 2}^{2\to 3} = 0.1$$

that is, if the defender is on the train from station 3 to 2, then at the next decision phase: she has probability 0.1 to choose Stay, thereby exiting the train and remaining at station 2; 0.8 to Go $2 \to 1$*, meaning she remains on her current train as it travels to station 1; and 0.1 to Go* $2 \to 3$*, meaning she exits her current train and boards the train heading the opposite direction toward station 3.*

Given π, the defender's movement is a Markov chain over the locations with *defender transition matrix* T_d, whose entry at column k, row l specifies the probability of a defender currently at location k being at location l during the next time step. In T_d, index i ($i \in 1, \ldots, N$) represents station i; indexes larger than N represent trains.

Example 2: *For Example 1,* T_d *is as follows:*

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 1\to 2 \\ 1\to 3 \\ \cdots \end{array} \begin{array}{c} \begin{array}{ccccc} 1 & 2 & \cdots 2\to 3 & 3\to 1 & 3\to 2 \end{array} \\ \begin{pmatrix} s_1 & 0 & 0 & s_{3\to 1} & 0 \\ 0 & s_2 & 0 & 0 & s_{3\to 2} \\ 0 & 0 & s_{2\to 3} & 0 & 0 \\ g_1^{1\to 2} & 0 & 0 & g_{3\to 1}^{1\to 2} & 0 \\ g_1^{1\to 3} & 0 & 0 & g_{3\to 1}^{1\to 3} & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots \end{pmatrix} \end{array}$$

π	Defender's Markov strategy	$\mathbf{c}_b^s$	Criminal's belief of $\mathbf{c}^s$
T_d	Defender transition matrix	$\mathbf{c}_b^t$	Criminal's belief of $\mathbf{c}^t$
$\mathbf{c}^s$	Defender stationary coverage	T_{db}	Criminal's belief of T_d
$\mathbf{c}^t$	Defender coverage vector at time step t	E	Target expected value for criminals
T_s	Transition matrix for the OSG Markov chain	p	Criminal's next strike probability

Table 3.1: Notation used throughout this chapter.

Using T_d and $\mathbf{c}^t = (c_1, c_2, \cdots, c_N, c_{1\to 2}, \cdots)^T$, defined as the probability distribution of a defender's location at time t, we can calculate the coverage vector at time step $t_1 > t$ through the formula

$$\mathbf{c}^{t_1} = (T_d)^{t_1 - t} \cdot \mathbf{c}^t \tag{3.1}$$

We restrict each element in π to be strictly positive so that T_d is ergodic, meaning it is possible to eventually get from every location to every other location in finite time. For an ergodic T_d, based on Lemma 3.1.1, there is a unique *stationary coverage* $\mathbf{c}^s$, such that $T_d \cdot \mathbf{c}^s = \mathbf{c}^s$. The dependence of $\mathbf{c}^s$ on T_d and hence on π is shown in Fig. 3.2. The defender's initial coverage, $\mathbf{c}^1$, is set to $\mathbf{c}^s$ so that the criminal will face an invariant distribution whenever he enters the system. This invariant initial distribution is analogous to assuming that the defender patrols for a long time and becomes stable, but under our model, criminals can enter the system at any time.

Lemma 3.1.1. *(Fundamental Theorem of Markov Chains) For an ergodic Markov chain P, there is a unique probability vector $\mathbf{c}$ such that $P \cdot \mathbf{c} = \mathbf{c}$ and $\mathbf{c}$ is strictly positive.*

Proof. This is a very simple restatement of the property of ergodic Markov chain. [44] provides detailed proof. □

3.1.2 Modeling Opportunistic Criminals

Our model of the criminal consists of three components.

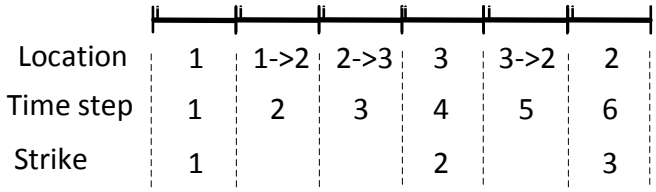


Figure 3.3: Example of strikes

Criminal's probability to commit a crime at the current time step: We assume the criminal will only commit crimes at stations, as discussed earlier [35], and only during action phases, since decision phases are considered instantaneous. The probability of such a crime is determined by two factors. The first is the *attractivenes* of each target station [51], which measures the availability of crime opportunities at a station. Attractiveness measures how likely a criminal located at that station during an action phase is to commit a crime *in the absence of defenders*; $\mathbf{Att} = (Att_1, Att_2, \cdots, Att_N)$ is the N vector composed of station attractiveness. The second factor is the defender's presence; i.e., if a criminal is at the same station as a defender, he will not commit a crime. Thus, his probability of committing a crime at station i will be influenced by $c^t(i)$. Using this strategy, the criminal will never be caught red handed by the defender, but may be forced toward a less attractive target. Thus, the probability of the criminal committing a crime if located at station i during the action phase of time step t, is denoted as $q_c(i,t) = (1 - c^t(i))Att(i)$.

Criminal's belief state of the defender: During the decision phase, the criminal decides the next target station; he then moves directly to that station at the next action phase(s). Hence, the criminal's motion within the metro system can be distilled down to a sequence of stations where he chooses to locate; we refer to these instances of attempted crime as *Strikes*. Figure 3 is a toy example showing the relationship between the time

steps and strikes for a criminal. As shown in the figure, only the time steps when the criminal is at stations are counted as strikes.

When making these target decisions, the criminal tends to choose stations with high *expected utilities*. He uses his knowledge of π and his real-time observations to make such decisions. Let T_{db}, $\mathbf{c}_b^t$, and $\mathbf{c}_b^s$ be his *belief* of T_d, $\mathbf{c}^t$, and $\mathbf{c}^s$, respectively. As the criminals have limited surveillance capability, these beliefs may not be the same as T_d, $\mathbf{c}^t$, and $\mathbf{c}^s$. To model the criminal's surveillance imperfection we use *anchoring bias* – a cognitive bias, with extensive experimental backing, which reveals the human bias toward choosing a uniform distribution when assigning probabilities to events under imperfect information [43, 49]. We denote the level of the criminal's anchoring bias with the parameter b, where $b = 0$ indicates no anchoring bias, and $b = 1$ indicates complete reliance on such bias. We set $T_{db} = (1 - b) \cdot T_d + b \cdot T_u$, with corresponding stationary coverage $\mathbf{c}_b^s$, where T_u corresponds to the uniform distribution.

At any given time step t when the criminal is at a station, i.e., a strike, he may be modeled as using his belief and observations to estimate $\mathbf{c}_b^t$. We assume the opportunistic criminal only uses his current observation, $\mathbf{c}_b^s$ and T_{db} to estimate $\mathbf{c}_b^t$ (criminal's belief of defender's location distribution). Specifically, if the criminal is at station i and the defender is also there, then $\mathbf{c}_b^t$ is $(0, 0, ..., 1, 0, ..., 0)^T$, where row i is 1 and all others are 0. Otherwise the defender is not at i, and

$$\mathbf{c}_b^t = \frac{(c_b^s(1), c_b^s(2), ..., 0, c_b^s(i+1), ..., c_b^s(N_l))^T}{\left[1 - c_b^s(i)\right]}, \tag{3.2}$$

where row i is 0 and other rows are proportional to the corresponding rows in $\mathbf{c}_b^s$. Our approach to compute $\mathbf{c}_b^t$ is justified on two grounds. First, it is computationally cheap. Second, as we show in experimental results, even perfect knowledge provides very limited improvement in the criminal's performance given our modeling of the criminal's bounded rationality and anchoring bias; thus a more complex procedure is unnecessary. Given $\mathbf{c}_b^t$ and T_{db}, the belief coverage vector at time step t_1 ($t_1 > t$), $\mathbf{c}_b^{t_1}$, is calculated via Eq. 3.1.

Input: i: the criminal's station; π: defender's Markov strategy; m: the defender's location; b: parameter of criminal's anchoring bias
Output: $p(\cdot|i, \mathbf{c}_b^{t_0})$: The criminal's probability distribution for next target

Initial N with the number of stations ;
Initial T_d by π;
Initial $\mathbf{c}^s$ with stationary coverage of T_d;
Initial $\mathbf{c}_b^{t_0}$ with a $1 \times (3N-2)$ zero vector ;
$T_{db} = (1-b) \cdot T_d + b \cdot T_u$;
$\mathbf{c}_b^s = (1-b) \cdot \mathbf{c}^s + b \cdot \mathbf{c}_u^s$;
if $i == m$ **then**
    $\mathbf{c}_b^{t_0}(i) = 1$;
end
if $i \neq m$ **then**
    for $j \in Location$ **do**
        $\mathbf{c}_b^{t_0}(j) = \dfrac{\mathbf{c}_b^s(j)}{1 - \mathbf{c}_b^s(i)}$;
    end
    $\mathbf{c}_b^{t_0}(i) = 0$;
end
for $j \in Station$ **do**
    $t = |i - j| + 1$;
    $\mathbf{c}_b^{t_0+t} = (T_{db})^t \cdot \mathbf{c}_b^{t_0}$;
    $E(j|i, \mathbf{c}_b^{t_0}) = \dfrac{\left(1 - \mathbf{c}_b^{t_0+t}(j)\right) Att(j)}{t}$;
end
for $j \in Station$ **do**
    $p(j|i, \mathbf{c}_b^{t_0}) = \dfrac{E(j|i, \mathbf{c}_b^{t_0})^\lambda}{\sum_{h=1}^{N} E(h|i, \mathbf{c}_b^{t_0})^\lambda}$;
end
return $p(\cdot|i, \mathbf{c}_b^{t_0})$;

Algorithm 1: Biased Random Walk Algorithm

We set the actual payoff for a crime to 1, but this can be generalized. The expected payoff for the criminal when choosing station j as the next strike, given that the current strike is at station i at time step t, is $q_{cb}(j, t + \delta_{ij})$, where $\delta_{ij} \geq 1$ is the minimum time needed to arrive at j from i. But, criminals are known to discount more distant locations when choosing targets. Therefore, the *utility* that the criminal places on a given payoff is discounted over time. We implement this by dividing the payoff by the time taken. Finally, the criminal must rely on his belief of the defender's coverage when evaluating $q_{cb}(j, t + \delta_{ij})$. Altogether, station j has the expected utility $E(j|i, \mathbf{c}_b^t) = \frac{q_{cb}(j,t+\delta_{ij})}{\delta_{ij}}$, which is

$$E(j|i, \mathbf{c}_b^t) = \frac{\left(1 - \left[(T_{db})^{\delta_{ij}} \cdot \mathbf{c}_b^t\right](j)\right) Att(j)}{\delta_{ij}}. \tag{3.3}$$

The criminal's Quantal Biased Random Movement (QBRM): Finally, we propose QBRM to model the criminal's bounded rationality based on other such models of criminal movements in urban domains [51]. Instead of always picking the station with highest expected utility, his movement is modeled as a random process biased toward stations of high expected utility. Given the expected value for each station $\mathbf{E}(\cdot|i, \mathbf{c}_b^t)$, the probability distribution for each being chosen as the next strike, $\mathbf{p}(\cdot|i, \mathbf{c}_b^t)$ is:

$$p(j|i, \mathbf{c}_b^t) = \frac{E(j|i, \mathbf{c}_b^t)^\lambda}{\sum_{h=1}^{N} E(h|i, \mathbf{c}_b^t)^\lambda}, \tag{3.4}$$

where $\lambda \geq 0$ is a parameter that describes the criminal's level of rationality. This is an instance of the *quantal response* model of boundedly rational behavior [37]. The criminal may, as an alternative to choosing a further strike, leave the system at *exit*

rate α. Therefore, the criminal eventually leaves the system with probability 1, and in expectation receives a finite utility; he cannot indefinitely increase his utility.

Given the criminal's QBRM, the Opportunistic Security Game can be simplified to a Stackelberg game for specific value of the parameters describing criminal's behaviour (Theorem 3.1.2).

Lemma 3.1.2. *When the criminal's rantionality level parameter* $\lambda = 0$*, the defender's optimal strategy is a stationary strategy, meaning that the defender picks a station and does not move in the patrol.*

Proof. According to Eqn. 3.4, when $\lambda = 0$, $p(j|i, \mathbf{c}_b^t) = \frac{1}{N}$ for all targets, which is independent of defender's Markov strategy π. Therefore, the OSG is equivalent to a Stackelberg Game where the leader (the criminal) makes his choice first, which is independent of the follower's (defender's) choice. Then the follower can decide her action given the leader's action. Therefore, as in a Stackelberg game, the follower's (defender's) optimal strategy is a pure strategy. Furthermore, we know that in this Stackelberg game, the leader (the criminal) is making a uniform random choice, meaning that he chooses each target with the same probability. Therefore, the defender's optimal strategy is staying at the station with highest attractiveness. □

To summarize, as shown in Figure 3.2, the opportunistic criminal is modeled as follows: First, he decides whether to commit a crime or not based on the defender's presence at his station at each strike. Next, he uses his imperfect belief T_{db} of the defender's strategy, which is affected by anchoring bias, and his real-time observation to update his belief $\mathbf{c}_b^t$ using a simple scheme (Eq. 3.2). Finally, we use QBRM to model his next attack

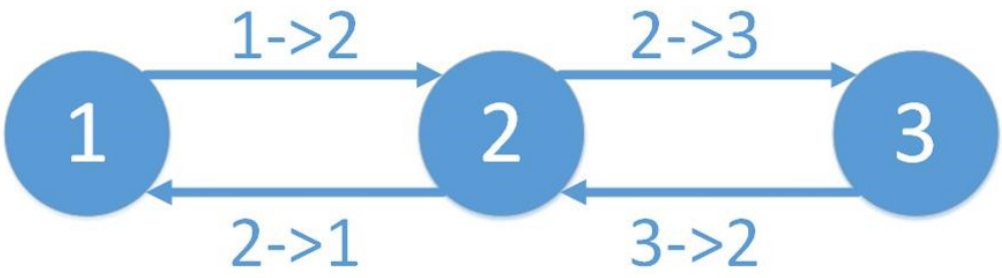
(Eq. 3.4) based on the expected utility of different targets (Eq. 3.3). Algorithm 1 is a full mathematical description of the criminal's movement. In Algorithm 1, steps 1-4 are initialization; steps 5-6 model how the criminal generates his imperfect belief; steps 7-15 model how the criminal updates his belief given his real-time observation; steps 16-20 model how the criminal evaluates each station based on his updated belief; and steps 21-24 use QBRM to model his probability distribution of visiting each station in his next strike.

3.2 Example of OSG

In this section, we are going to show an toy example of the OSG we discussed above. Figure 3.4(a) shows a transportation network with 3 nodes. Figure 3.4(b) is the game matrix when both defender and criminal are at station 2. The row represents the combination of criminal's crime decision and target decision while the column represents the defenders' strategy for next time step. Figure 3.4(c) shows another game matrix when criminal is on train from target 2 to target 1 while the defender is at target 2. The opportunistic crime is modeled as this game that has been played repeatedly.

3.3 Exact OSG (EOSG) algorithm

Given the defender and criminal models, the EOSG algorithm computes the optimal defender strategy by modeling the OSG as a finite state Markov chain. As all the criminals behave identically, we can focus on the interaction between the defender and one criminal without loss of generality.



(a) Transportation network

	Stay	Go 2->1	Go 2->3
Crime, Stay	(-1,1)	(-1,1)	(-1,1)
No, Stay	(0,0)	(0,0)	(0,0)
Crime, Go 2->1	(-1,1)	(-1,1)	(-1,1)
No, Go 2->1	(0,0)	(0,0)	(0,0)
Crime, Go 2->3	(-1,1)	(-1,1)	(-1,1)
No, Go 2->3	(0,0)	(0,0)	(0,0)

(b) Game matrix 1

	Stay	Go 2->1	Go 2->3
Crime, Get off	(1,-1)	(1,-1)	(1,-1)
No, Get off	(0,0)	(0,0)	(0,0)
Crime, Go 1->2	(1,-1)	(1,-1)	(1,-1)
No, Go 1->2	(0,0)	(0,0)	(0,0)

(c) Game matrix 2

Figure 3.4: Example of OSG

Each state of the EOSG Markov chain is a combination of the criminal's station and the defender's location. Here we only consider situations where the criminal is at a station as states because he only makes decisions at stations. Since there are N stations and N_l locations, the number of states is $N \cdot N_l$ in the EOSG markov chain. State transitions in this EOSG markov chain are based on *strikes* rather than *time steps*. The transition matrix for this Markov chain, denoted as T_s, can be calculated by combining the defender and criminal models. For further analysis, we pick the element $p_{S1\to S2}$ in T_s that represents the transition probability from state $S1$ to $S2$. Suppose in $S1$ the criminal is at station i while the defender is at location m at time step t, and in $S2$, the criminal is at station j while the defender is at location n at time step $t + \delta_{ij}$. We need two steps to calculate the transition probability $p_{S1\to S2}$. First, we find the transition probability of the criminal from i to j, $p(j|i, \mathbf{c}_b^t)$. Then, we find the defender's transition probability

from m to n, which is $c^{t+\delta_{ij}}(n) = \left((T_d)^{\delta_{ij}} \cdot \mathbf{e}_m\right)(n)$, where $\mathbf{e}_m$ is a basis vector for the current location m. The transition probability $p_{S1\to S2}$ is therefore given by

$$p_{S1\to S2} = p(j|i, \mathbf{c}_b^t) \cdot c^{t+\delta_{ij}}(n). \tag{3.5}$$

Since $p(j|i, \mathbf{c}_b^t)$ and $c^{t+\delta_{ij}}(n)$ are determined by π, $p_{S1\to S2}$ is also in terms of π (see Fig. 3.2), and hence so is T_s.

Given this EOSG model, we can calculate the defender's expected utility at each strike. For each successful crime, the defender receives utility $u_d < 0$, while if there is no crime, she receives utility 0. We do not consider the time discount factor in the defender's expected utility, as the goal of the defender shall be to simply minimize the total expected number of crimes that any criminal will commit. Formally, we define a vector $\mathbf{r}_d \in \mathbf{R}^{N \cdot N_l}$ such that entries representing states with both criminal and defender at the same station are 0 while those representing states with criminal at station i and defender not present are $Att(i) \cdot u_d$. Then, the defender's expected utility $V_d(t)$ during strike number t is $V_d(t) = \mathbf{r}_d \cdot \mathbf{x}_t$, where $\mathbf{x}_t$ is the state distribution at strike number t. $\mathbf{x}_t$ can be calculated from the initial state distribution $\mathbf{x}_1$, via $\mathbf{x}_t = ((1-\alpha) \cdot T_s)^{t-1}\mathbf{x}_1$. The initial state distribution $\mathbf{x}_1$ can be calculated from the initial criminal distribution and $\mathbf{c}^s$. The defender's total expected utility over all strikes is thus

$$\begin{aligned} Obj &= \lim_{\ell\to\infty} \sum\nolimits_{t=1}^{\ell} V_d(t) \\ &= \mathbf{r}_d \cdot (I - (1-\alpha)T_s)^{-1}\mathbf{x}_1 \ , \end{aligned} \tag{3.6}$$

where I is an identity matrix and α is the criminal's *exit rate*. In this equation we use the geometric sum formula and the fact that the largest eigenvalue of T_s is 1, so that $I - (1-\alpha)T_s$ is nonsingular for $0 < \alpha < 1$.

The objective is a function of the transition matrix T_s and $\mathbf{x}_1$, which can be expressed in terms of π via Eqs. (3.1), (3.3), (3.4), and (3.5). We have thus formulated the defender's problem of finding the optimal Markov strategy to commit to as a nonlinear optimization problem, specifically to choose π to maximize Obj (that is, minimize the total amount of crime).

3.4 OSG for multiple defenders

If K multiple defenders all patrol the entire metro, using the same π, which is denoted as *full length patrolling*, then they will often be at the same station simultaneously, which carries no benefit. On the other hand if we allow arbitrary defenders' strategies that are correlated, we will need to reason about complex *real-time* communication and coordination among defenders. Instead, we divide the metro into K contiguous segments, and designate one defender per segment, as in typical real-world patrolling of a metro system. Each defender will have a strategy specialized to her segment.

Defenders: In the k-th segment, the number of locations is n_l^k. Defender k patrols with the Markov strategy π_k. Her transition matrix is $T_{dk} \in \mathbf{R}^{n_l^k \times n_l^k}$. Her coverage vector at time t is $\mathbf{c}_k^t$, and $\mathbf{c}_k^s$ is her stationary coverage. Hence, defender k's behavior is the same as that in a single-defender OSG, while the collective defender behavior is described by the Markov strategy $\pi = (\pi_1, \pi_2, ..., \pi_K)$. The transition matrix T_d is as follows, where

Input: i: the criminal's station; π: vector of defender Markov strategies; $\mathbf{m}$: vector of defender locations; b: parameter of criminal's anchoring bias

Output: $p(\cdot|i, \mathbf{c}_b^{t_0})$: The criminal's probability distribution for next target

Initial N with the number of stations ;

Initial K with the number of defenders ;

Initial k_i with the segment that station i is in ;

for $k \leq K$ **do**

  Initial T_{dk} by π_k;

  Initial $\mathbf{c}_k^s$ by stationary coverage of T_{dk};

  $T_{dbk} = (1-b) \cdot T_{dk} + b \cdot T_{uk}$;

  $\mathbf{c}_{bk}^s = (1-b) \cdot \mathbf{c}_k^s + b \cdot \mathbf{c}_{uk}^s$;

  $\mathbf{c}_{bk}^{t_0} = \mathbf{c}_{bk}^s$

  if $k == k_i$ **then**

    Initial $\mathbf{c}_{bk}^{t_0}$ with a $1 \times n_l^k$ zero vector ;

    if $i == \mathbf{m}(k)$ **then**

      $\mathbf{c}_{bk}^{t_0}(i) = 1$;

    end

    if $i \neq \mathbf{m}(k)$ **then**

      for $j \in$ *Location in segment* k **do**

        $\mathbf{c}_{bk}^{t_0}(j) = \dfrac{\mathbf{c}_{bk}^s(j)}{1 - \mathbf{c}_{bk}^s(i)}$;

      end

      $\mathbf{c}_{bk}^{t_0}(i) = 0$;

    end

  end

end

$$T_{db} = \begin{pmatrix} T_{db1} & 0 & \dots & 0 \\ 0 & T_{db2} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & T_{dbK} \end{pmatrix}$$

$\mathbf{c}_b^{t_0} = (\mathbf{c}_{b1}^{t_0}; \mathbf{c}_{b2}^{t_0}; ...; \mathbf{c}_{bK}^{t_0})$.

for $j \in$ *Station* **do**

  $t = |i - j| + 1$;

  $\mathbf{c}_b^{t_0+t} = (T_{db})^t \cdot \mathbf{c}_b^{t_0}$;

  $E(j|i, \mathbf{c}_b^{t_0}) = \dfrac{\left(1 - \mathbf{c}_b^{t_0+t}(j)\right) Att(j)}{t}$;

end

for $j \in$ *Station* **do**

  $p(j|i, \mathbf{c}_b^{t_0}) = \dfrac{E(j|i, \mathbf{c}_b^{t_0})^\lambda}{\sum_{h=1}^N E(h|i, \mathbf{c}_b^{t_0})^\lambda}$;

end

return $p(\cdot|i, \mathbf{c}_b^{t_0})$;

Algorithm 2: Biased Random Walk Algorithm with Mulitple defenders

we have dropped the trains between segments from the basis for T_d and ensured that station numbering is continuous within segments:

$$T_d = \begin{pmatrix} T_{d1} & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & T_{dK} \end{pmatrix}. \tag{3.7}$$

The coverage of all units at time step t is $\mathbf{c}^t$, and is defined as the concatenation of coverage vectors $(\mathbf{c}_1^t; \mathbf{c}_2^t; ...; \mathbf{c}_K^t)$. $\mathbf{c}^t$ sums to K since each $\mathbf{c}_k^t$ sums to 1. The vector $\mathbf{c}^t$ evolves to future time steps t_1 in the same way as before, via Eq. 3.1. The overall stationary coverage is $\mathbf{c}^s = (\mathbf{c}_1^s; \mathbf{c}_2^s; ...; \mathbf{c}_K^s)$.

Opportunistic criminals: The previous model for criminals still applies. However, any variables related to defenders (T_d, $\mathbf{c}^t$, $\mathbf{c}^s$) are replaced by their counterparts for the multiple defenders. Furthermore, the criminal in segment k at time t cannot observe defenders other than k. As a result, his belief of defender coverage is updated only for segment k, i.e., $\mathbf{c}_b^t = (\mathbf{c}_{b1}^s; \mathbf{c}_{b2}^s; ...; \mathbf{c}_{b(k-1)}^s; \mathbf{c}_{bk}^t; \mathbf{c}_{b(k+1)}^s; ...; \mathbf{c}_{bK}^s)$. Algorithm 2 describes a criminal's behavior in the multiple defenders settings. Similar to Algorithm 1, in Algorithm 2, steps 1-3 are initialization; steps 4-22 model how the criminal generates and updates his imperfect belief for each defender, such that for defender $k(k \leq K)$, the process of calculating the criminal's belief is exactly the same as the single defender scenario; steps 23-24 combine the criminal's belief for each defender as his belief for all the defenders; steps 25-29 model how the criminal evaluates each station based on his belief; and steps 30-34 use QBRM to model his probability distribution of visiting each station in his next strike.

EOSG: In optimizing defender strategies via a Markov chain, each state records the station of the criminal and the location of *each* defender. As a result, each state is denoted as $S = (i, m_1, ..., m_K)$, where the criminal is at station i and defender k is at location m_k. Since defender k can be at n_l^k different locations, the total number of states is $N \cdot n_l^1 \cdots n_l^K$. To apply EOSG for multiple defenders, T_s is still calculated using the defender and criminal models. The transition probability $p_{S_1 \to S_2}$ from $S_1 = (i, m_1, ..., m_K)$ at time t to $S_2 = (j, n_1, ..., n_K)$ at time $t + \delta_{ij}$ is

$$p_{S_1 \to S_2} = p(j|i, \mathbf{c}_b^t) \prod\nolimits_k c^{t+\delta_{ij}}(n_k),$$

where $c^{t+\delta_{ij}}(n_k) = ((T_d)^{\delta_{ij}} \cdot \mathbf{e}_{m_1,m_2,...,m_K})(n_k)$ and $\mathbf{e}_{m_1,m_2,...,m_K}$ is an indicator vector with 1 at entries representing locations $m_1, m_2, ..., m_K$ and 0 at all others. The state distribution $\mathbf{x}$ and revenue $\mathbf{r}_d$ are both $N \cdot n_l^1 \cdots n_l^K$ vectors. The defenders' total expected utility is given by Eq. (3.6); our goal remains to find a π to maximize Obj.

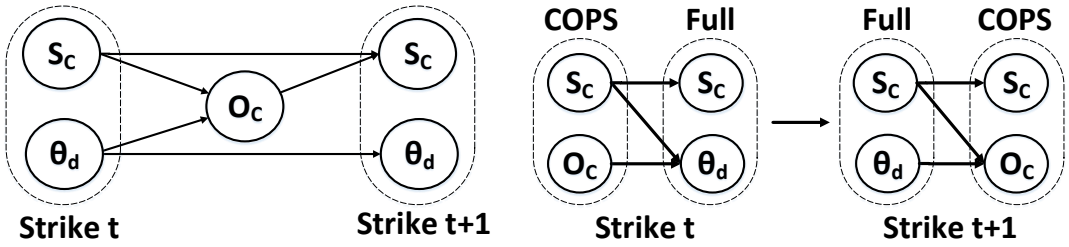
3.5 The COPS Algorithm

The objective of EOSG can be formulated as a non-linear optimization. Unfortunately, as we will show in our experiments, the EOSG algorithm fails to scale-up to real-world sized problem instances due to the size of T_s in Eq. (3.6), which is exponential ($N \cdot n_l^1 \cdots n_l^K$ by $N \cdot n_l^1 \cdots n_l^K$) for K defenders. We propose the Compact OPportunistic Security game state (COPS) algorithm to accelerate the computation. COPS simplifies the model by compactly representing the states. The size of the transition matrix in COPS is $2N \times 2N$, *regardless of the number of defenders*, which is dramatically smaller than in the

exact algorithm. The COPS algorithm is inspired by the Boyen-Koller(BK) algorithm for approximate inference on Dynamic Bayesian Networks [12]. COPS improves upon a direct application of BK's factored representation by maintaining strong correlations between locations of players in OSG.

In OSG with a single defender, there are two components in a Markov chain state for strike t: the station of the criminal S_c^t and the location of the defender θ_d^t. These two components are correlated when they evolve. We introduce an intermediate component, the criminal's observation O_c^t, which is determined by both S_c^t and θ_d^t. Given the criminal's current station and his observation, we can compute his distribution over the next strike station. At the same time, the evolution of θ_d^t is independent of S_c^t. Such evolution is shown in Figure 3.5(a). This is an instance of a Dynamic Bayesian Network: S_c^t, O_c^t, and θ_d^t are the random variables, while edges represent probabilistic dependence.

A direct application of the Boyen-Koller algorithm compactly represents the states by using the marginal distribution of these two components, S_c^t and θ_d^t, as approximate states. The marginal distributions of S_c^t and θ_d^t are denoted as $\Pr(S_c^t)$ and $\Pr(\theta_d^t)$ respectively, and it is assumed that these two components are independent, meaning we can restore the Markov Chain states by multiplying these marginal distributions. Note that in Section 4.2, we set $\Pr(\theta_d^t) = \mathbf{c}^s$ for all strikes. Thus, we do not need to store θ_d^t in the state representation. Therefore, the total number of the approximate states in this case is just N. However, such an approximation throws away the strong correlation between the criminal's station and defender unit's location through the criminal's real-time observation. Our preliminary experiments showed that this approximate algorithm leads to low defender expected utility.



(a) Full state evolution (b) COPS state evolution

Figure 3.5: COPS algorithm

To design a better algorithm, we should add more information about the correlation between the criminal and defenders. To that end, our COPS algorithm compactly represents our Markov Chain states with less information lost. Instead of just considering the marginal distributions of each component $\Pr(\theta_d^t)$ and $\Pr(S_c^t)$, we also include the observation of the criminal O_c^t while constructing the approximate states. The criminal's observation is binary: 1 if the defender is at the same station with him, 0 otherwise. The new approximate states, named *COPS states*, only keep the marginal probability distribution of $\Pr(S_c^t, O_c^t)$. So, the new state space is the Cartesian product of the sets of S_c^t and O_c^t, which has size $2N$.

One subtask of COPS is to recover the distributions over the full state space (S_c^t, θ_d^t), given our state representation $\Pr(S_c^t, O_c^t)$. We cannot restore such distribution by multiplying $\Pr(\theta_d^t)$ and $\Pr(S_c^t)$ in COPS. This is because S_c^t, O_c^t, and θ_d^t are not independent. For example, in COPS state $S_c^t = 1$, $O_c^t = 1$, θ_d^t cannot be any value except 1. In other words, the defender's location distribution $\Pr(\theta_d^t | S_c^t, O_c^t)$ is no longer $\mathbf{c^s}$. Instead, we approximate $\Pr(\theta_d^t | S_c^t, O_c^t)$ as follows. In each COPS state (S_c^t, O_c^t), the *estimated marginal distribution* for the defender, $\widehat{\Pr}(\theta_d^t | S_c^t, O_c^t)$, is found in a manner similar to that used to find the criminal's belief distribution $\mathbf{c}_b^t$. Specifically, if $O_c^t = 1$, $\widehat{\Pr}(\theta_d^t | S_c^t, O_c^t) = (0, 0, ..., 1, 0, ..., 0)^T$, where the row representing station S_c^t is 1 and all

others are 0; if $O_c^t = 0$, then $\widehat{\Pr}(\theta_d^t | S_c^t, O_c^t)$ is found through Equation 3.2, but with the $c_b^s(j)$ replaced by the true stationary coverage value $c^s(j)$. We can then recover the estimated distribution over full states $\widehat{\Pr}(S_c^t = i, \theta_d^t | S_c^t = i, O_c^t) = \widehat{\Pr}(\theta_d^t | S_c^t = i, O_c^t)$ for all i and $\widehat{\Pr}(S_c^t = j, \theta_d^t | S_c^t = i, O_c^t) = 0$ for all $j \neq i$. Estimated full distributions evolve the same way as exact distributions do, as described in Section 4. At the future strike, we can then project the evolved estimated full distribution to distributions over COPS states. Figure 3.5(b) shows the whole process of the evolution of COPS states. However, such a process would appear to involve representing a full T_s, negating the benefit of the factored representation; we avoid that by using T_{COPS}, discussed below.

To streamline the process of evolving COPS states, in practice we use a transition matrix $T_{COPS} \in \mathbf{R}^{2N \times 2N}$. Each element of T_{COPS}, i.e., transition probability $\Pr(S_c^{t'}, O_c^{t'} | S_c^t, O_c^t)$, can be calculated as follows:

$$\begin{aligned}
&\Pr(S_c^{t'}, O_c^{t'} | S_c^t, O_c^t) \\
&= \sum\nolimits_{\theta_d^{t'}} \sum\nolimits_{\theta_d^t} \Pr(S_c^{t'}, O_c^{t'} | S_c^{t'}, \theta_d^{t'}) \cdot \Pr(S_c^{t'}, \theta_d^{t'} | S_c^t, \theta_d^t) \cdot \widehat{\Pr}(S_c^t, \theta_d^t | S_c^t, O_c^t) \\
&= \Pr(S_c^{t'} | S_c^t, O_c^t) \sum\nolimits_{\theta_d^{t'}} \Pr(O_c^{t'} | S_c^{t'}, \theta_d^{t'}) \cdot \sum\nolimits_{\theta_d^t} \Pr(\theta_d^{t'} | S_c^{t'}, S_c^t, \theta_d^t) \cdot \widehat{\Pr}(\theta_d^t | S_c^t, O_c^t),
\end{aligned} \tag{3.8}$$

where $\Pr(S_c^{t'} | S_c^t, O_c^t)$ and $\Pr(\theta_d^{t'} | S_c^{t'}, S_c^t, \theta_d^t)$ correspond to $p(j|i, \mathbf{c}_b^{t_0})$ and $c^{t_0+|i-j|+1}(n)$, respectively, in Section 4.

The defenders' total expected utility in COPS is calculated in a similar way as the exact algorithm, which is

$$Obj_{COPS} = \mathbf{r}_{d,COPS} \cdot (I - (1-\alpha) T_{COPS})^{-1} \mathbf{x}_{1,COPS}, \tag{3.9}$$

where $\mathbf{r}_{d,COPS}$, $\mathbf{x}_{1,COPS}$ are the expected utility vector and the initial distribution for COPS states. Similar to $\mathbf{r}_d$, $\mathbf{r}_{d,COPS}(S)$ is 0 if in state S the defender is at the same station with the criminal, else $\mathbf{r}_{d,COPS}(S) = u_d$. COPS is faster than the exact algorithm because the number of states is reduced dramatically. Meanwhile, the approximation error of COPS algorithm is bounded according to Theorem 3.5.2.

Definition 3.5.1. *Let m_i be the location corresponding to station i. For a distribution over OSG full states $\mathbf{x}$, the corresponding distribution over COPS states $\mathbf{x}_{COPS}$ is:*

$$\mathbf{x}_{COPS}(i,o) = \begin{cases} \mathbf{x}(i,m_i) & \text{if } o = 1 \\ \sum_{m \neq m_i} \mathbf{x}(i,m) & \text{if } o = 0 \end{cases}$$

For a distribution over COPS states $\mathbf{x}_{COPS}$, the corresponding approximate distribution over OSG full states $\mathbf{x}'$ is:

$$\mathbf{x}'(i,m) = \begin{cases} \mathbf{x}_{COPS}(i,1) & \text{if } m = m_i \\ \mathbf{x}_{COPS}(i,0) \cdot \frac{c^s(m)}{1-c^s(i)} & \text{otherwise} \end{cases}$$

This conversion can be summarized through a single matrix multiplication, such that $\mathbf{x}' = A\mathbf{x}$.

Lemma 3.5.1. *Let μ_2 be the magnitude of the second largest eigenvalue of transition matrix T_s. Let δ be the largest possible L_2 approximation error introduced when full state distribution $\mathbf{x}$ is transformed into the COPS representation vector $\mathbf{x}_{COPS}$ and back into the approximate distribution $\mathbf{x}'$ over full states: $||\mathbf{x} - A\mathbf{x}|| \leq \delta$. At strike number t, the L_2 norm between the EOSG distribution $\mathbf{y}_t$ and the distribution found through COPS algorithm $\mathbf{x}_t$ is bounded, such that $||\mathbf{y}_t - \mathbf{x}_t||_2 \leq (1-\alpha)^{t-1} \frac{\delta(1-\mu_2^t)}{1-\mu_2}$.*

Proof. Let $\mathbf{x}_t$ be the state vector as found through the COPS algorithm at time t. The time evolution for $\mathbf{x}$ proceeds then as follows: $\mathbf{x}_t = (1-\alpha)^{t-1}(AT_s)^{t-1}\mathbf{x}_1$, where $\mathbf{x}_1 = A\mathbf{y}_1$,

and $\mathbf{y}_1$ is the initial state vector for the EOSG algorithm. So, consider the L_2 error introduced at iteration t by the COPS approximation alone

$$||T_s\mathbf{x}_t - AT_s\mathbf{x}_t||_2 = (1-\alpha)^{t-1}||T_s(AT_s)^{t-1}\mathbf{x}_1 - AT_s(AT_s)^{t-1}\mathbf{x}_1||_2.$$

Since the vector $T_s(AT_s)^{t-1}\mathbf{x}_1$ is a full state vector, the error bound here is simply

$$||T_s\mathbf{x}_t - AT_s\mathbf{x}_t||_2 \leq \delta(1-\alpha)^{t-1}. \tag{3.10}$$

Now, assume that the error between the state vectors $\mathbf{x}_t$ and $\mathbf{y}_t$ at some time t is bound by ϵ: $||\mathbf{y}_t - \mathbf{x}_t||_2 \leq \epsilon$. Since in the EOSG Markov chain it is possible to travel from any state to any other state in a finite amount of time, this Markov chain is ergodic. Let the stationary distribution of T_s be $\mathbf{x^s}$, which is normalized such that $\overrightarrow{1} \cdot \mathbf{x}^s = 1$. $\mu_1 = 1 > \mu_2 \geq ... \geq \mu_{N \cdot N_l}$ are the magnitudes of the eigenvalues of T_s corresponding to eigenvectors $v_1(= \mathbf{x^s}), v_2, ..., v_{N \cdot N_l}$. Since T_s is the transition matrix of an ergodic Markov chain, $\mu_k < 1$ for $k \geq 2$. For eigenvectors $v_k, k \geq 2$, we have $|T_s \cdot v_k| = |\mu_k \cdot v_k|$. Multiplying by $\overrightarrow{1}$ and noting that $\overrightarrow{1} \cdot T_s = \overrightarrow{1}$, we get $|\overrightarrow{1} \cdot v_k| = |\mu_k \cdot \overrightarrow{1} \cdot v_k|$. Since $\mu_k \neq 1$, $\overrightarrow{1} \cdot v_k = 0$.

Write $\mathbf{x}_t$ and $\mathbf{y}_t$ in terms of $v_1, v_2, ..., v_{N \cdot N_l}$ as:

$$\mathbf{y}_t = \beta_1\mathbf{x^s} + \sum_{i=2}^{N \cdot N_l} \beta_i v_i$$
$$\mathbf{x}_t = \beta'_1\mathbf{x^s} + \sum_{i=2}^{N \cdot N_l} \beta'_i v_i$$

Since $\mathbf{y}_t = (1-\alpha)^{t-1} T_s^{t-1} \mathbf{y}_1$, then $\overrightarrow{1} \cdot \mathbf{y}_t = (1-\alpha)^{t-1}$; similarly, $\overrightarrow{1} \cdot \mathbf{x}_t = (1-\alpha)^{t-1}$. Multiplying both equations above by $\overrightarrow{1}$, we get $\beta_1 = \beta_1' = (1-\alpha)^{t-1}$. Therefore,

$$\begin{aligned} ||T_s \cdot \mathbf{y}_t - T_s \cdot \mathbf{x}_t||_2 &\leq || \sum_{i=2}^{N \cdot N_l} (\beta_i - \beta_i') \mu_i v_i ||_2 \\ &\leq |\mu_2| \sqrt{(\beta_2 - \beta_2')^2 + (\beta_3 - \beta_3')^2 + \cdots + (\beta_{N \cdot N_i} - \beta_{N \cdot N_i}')^2} \\ &\leq \mu_2 ||\mathbf{x}_t - \mathbf{y}_t||_2 \\ &\leq \mu_2 \epsilon \end{aligned}$$

Accordingly, at $t = 1$, we have

$$||\mathbf{y}_1 - \mathbf{x}_1||_2 = ||\mathbf{y}_1 - A\mathbf{y}_1||_2 \leq \delta \ .$$

At $t = 2$, we have

$$\begin{aligned} ||\mathbf{y}_2 - \mathbf{x}_2||_2 = (1-\alpha)||T_s \mathbf{y}_1 - A T_s \mathbf{x}_1|| = (1-\alpha)||T_s \mathbf{y}_1 - A T_s \mathbf{x}_1 + T_s \mathbf{x}_1 - T_s \mathbf{x}_1|| \leq \\ (1-\alpha)||T_s \mathbf{y}_1 - T_s \mathbf{x}_1||_2 + (1-\alpha)||T_s \mathbf{x}_1 - A T_s \mathbf{x}_1||_2 \ . \end{aligned}$$

From above, the bound for the first term is $\mu_2 \delta$, given the error bound at $t = 1$. The bound for the second term is directly given by (3.10), and is simply δ. Hence

$$||\mathbf{y}_2 - \mathbf{x}_2||_2 \leq (1-\alpha)\delta(\mu_2 + 1) \ .$$

At $t = 3$, we have

$$||\mathbf{y}_3 - \mathbf{x}_3||_2 = (1-\alpha)||T_s\mathbf{y}_2 - AT_s\mathbf{x}_2|| = (1-\alpha)||T_s\mathbf{y}_2 - AT_s\mathbf{x}_2 + T_s\mathbf{x}_2 - T_s\mathbf{x}_2|| \leq$$

$$(1-\alpha)||T_s\mathbf{y}_2 - T_s\mathbf{x}_2||_2 + (1-\alpha)||T_s\mathbf{x}_2 - AT_s\mathbf{x}_2||_2 \ .$$

From above, the bound for the first term is $\mu_2(1-\alpha)\delta(\mu_2+1)$, given the error bound at $t = 2$. The bound for the second term is taken from (3.10), and is $\delta(1-\alpha)$. Hence

$$||\mathbf{y}_3 - \mathbf{x}_3||_2 \leq (1-\alpha)^2\delta(\mu_2^2 + \mu_2 + 1) \ .$$

By extension, then, the error bound at time step t between EOSG and COPS states is:

$$||\mathbf{y}_t - \mathbf{x}_t||_2 \leq (1-\alpha)^{t-1}\delta\sum_{i=0}^{t-1}\mu_2^i = (1-\alpha)^{t-1}\delta\ \frac{1-\mu_2^t}{1-\mu_2} \ .$$

□

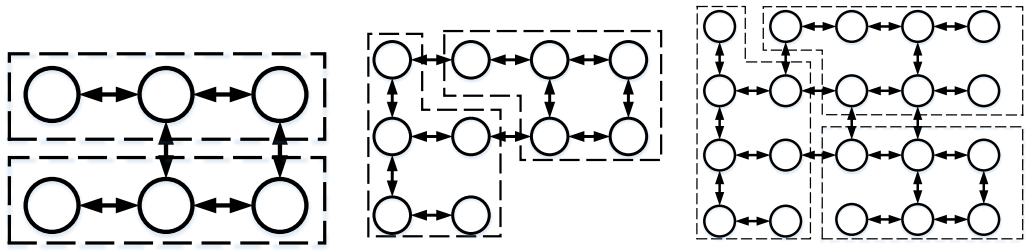
Theorem 3.5.2. *The difference between the EOSG objective and the COPS approximate objective* $|Obj - Obj_{COPS}|$ *is bounded by* $\frac{\sqrt{N \cdot N_l}\delta|u_d|}{[1-(1-\alpha)\mu_2]\ \alpha}$

Proof. Since Lemma 3.5.1 gives the bound of L_2 distance while $|Obj - Obj_{COPS}|$ is L_1 distance, we use the fact that for any two vectors v_1, v_2, the relationship between the L_1 distance and L_2 distance is: $||v_1 - v_2||_2 \leq ||v_1 - v_2||_1 \leq \sqrt{n}||v_1 - v_2||_2$, where n is the dimension of the vectors. Therefore, $||\mathbf{y}_t - \mathbf{x}_t||_1 \leq \frac{\sqrt{N \cdot N_l}(1-\alpha)^{t-1}(1-\mu_2^t)\delta}{1-\mu_2}$. Hence we have:

$$
\begin{aligned}
|Obj - Obj_{COPS}| &= \sum_{t=1}^{\infty} |\mathbf{r}_d \cdot \mathbf{y}_t - \mathbf{r}_d \cdot \mathbf{x}_t| \\
&= \sum_{t=1}^{\infty} |\mathbf{r}_d \cdot (\mathbf{y}_t - \mathbf{x}_t)| \\
&\leq \sum_{t=1}^{\infty} |r_{max}| ||\mathbf{y}_t - \mathbf{x}_t||_1 \\
&\leq |r_{max}| \sum_{t=1}^{\infty} \frac{\sqrt{N \cdot N_l}(1-\alpha)^{t-1}(1-\mu_2^t)\delta}{1-\mu_2} \\
&= |r_{max}| \frac{\sqrt{N \cdot N_l}\ \delta}{[1-(1-\alpha)\mu_2]\ \alpha}
\end{aligned}
$$

where r_{max} is the element in $\mathbf{r}_d$ with largest magnitude, which is $min(Att(i) \cdot u_d)$ because $\mathbf{r}_d$ is a non-positive vector by definition. Given $Att(i) \leq 1$, we have $|Obj - Obj_{COPS}| \leq \frac{\sqrt{N \cdot N_l}\delta |u_d|}{[1-(1-\alpha)\mu_2]\ \alpha}$ □

3.6 Experimental Results



(a) 6 stations (b) 10 stations (c) 20 stations

Figure 3.6: Part of metro systems in mega cities

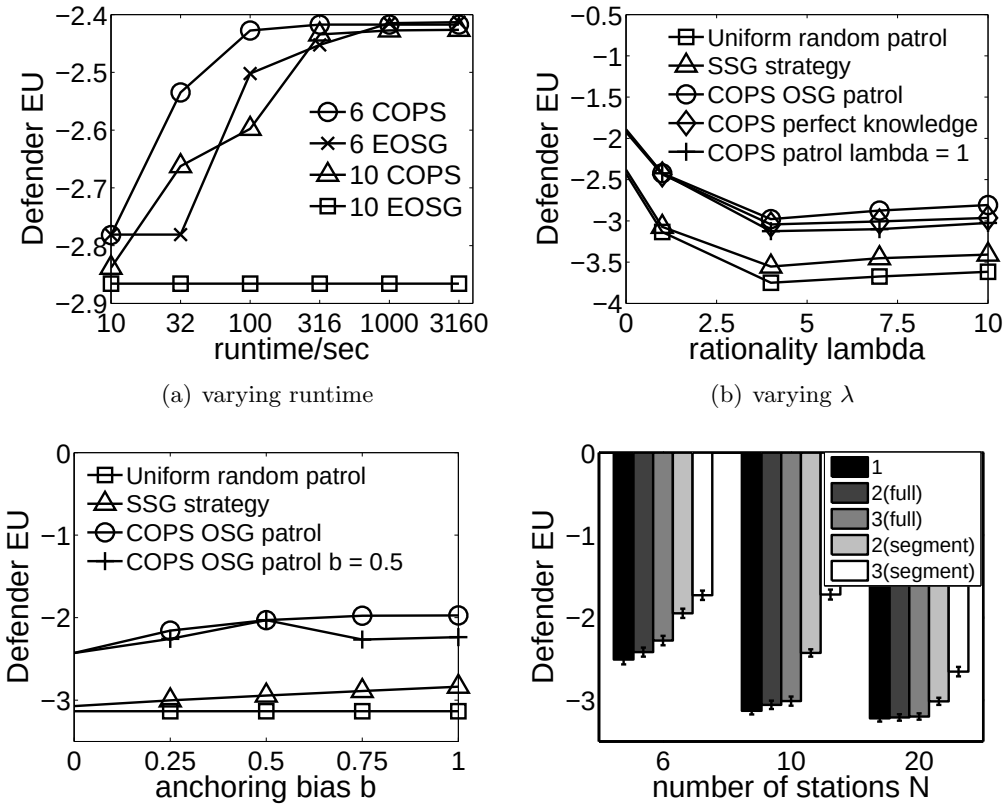
Settings: We use the graphs in Figure 3.6 – metro structures commonly observed in the world's mega cities – in our experiments. We also tested our algorithm on line structure systems, and the results are similar. We solve the non-linear optimization in

OSG using the `FindMaximum` function in Mathematica, which computes a locally optimal solution using an Interior Point algorithm. Each data point we report is *an average of 30 different instances*, each based on a different attractiveness setting; these instances were generated through a uniform random distribution from 0 to 1 for the attractiveness of each station. For multiple patrol unit scenarios, we use segment patrolling (except for Fig. 3.7(d)), and divide the graph so that the longest distances in each segments are minimized; the dashed boxes in Fig. 3.6 show the segments used. Results for other segmentations are similar. The defender's utility of a successful crime is $u_d = -1$. The criminal's initial distribution is set to a uniform distribution over stations. The criminal exit rate is $\alpha = 0.1$. Strategies generated by all algorithms are evaluated using Equation 3.6. All key results are *statistically significant* ($p < 0.01$).

Results: Fig. 3.7(a) shows the performance of the COPS algorithm and the EOSG algorithm using the settings from Fig. 3.6(a) and Fig. 3.6(b). In both, we set $\lambda = 1$. The Interior Point algorithm used by Mathematica is a locally optimal solver and there is always a current best feasible solution available, although the quality of the solution keeps improving through iterations. Therefore, one practical way to compare solutions is to check the solution quality after a fixed run-time. The x-axis in this figure shows runtime in seconds on a log scale, while the y-axis maps the defenders' average expected utility against one criminal, achieved by the currently-best solution at a given run time. Focusing first on results of 6 stations, where we have one defender, COPS outperforms EOSG for any runtime within 100 s, even though COPS is an approximate algorithm. This is because COPS reaches a local optimum faster than EOSG. Further, even for runtime long enough for EOSG to reach its local optimum (3160 s), where it outperforms

COPS, the difference in solution quality is less than 1%. Focusing next on results of 10 stations with 2 defenders (using segment patrolling), the conclusions are similar to 6 stations, but the advantage of COPS is more obvious in this larger scale problem. In most instances, COPS reaches a local optimum in 1000 s while the output of EOSG are the same as initial values in 3160 s.

Figure 3.7(b) employs criminals with varying levels of rationality to compare the performance of three different strategies: the uniform random strategy, which is a Markov strategy with equal probability for all available actions at each location; an SSG strategy, which is the optimal strategy against a strategic attacker that attacks a single target; and a COPS OSG strategy (given 1800 s so it reached a local optimum). In Fig. 3.7(b), we set $b = 0$; results with other b are similar. The system consists of 10 stations and 2 defenders. The COPS OSG strategy outperforms the random and SSG strategies significantly for any λ. Next, two more settings are tested: the first is the OSG strategy against criminals who have perfect knowledge of defenders' current location. This is a purely hypothetical setting, and created only to check if a more complex criminal belief model than the one in Eq. 3.2 would have led to significantly different defender performance. The degradation in performance against perfect criminals is less than 6%, indicating that a more complex belief update for defenders' current location would have insignificant impact on the results. The second is also an OSG strategy, but the defenders set a fixed λ during computation to test performance when the defender has an inaccurate estimate of λ. We picked $\lambda = 1$ from a set of sampled λ, since the OSG strategy with $\lambda = 1$ performs best against criminals with various levels of rationality. Even though the OSG strategy assuming $\lambda = 1$ performs slightly worse than that using the correct λ, it is still better than SSG



(a) varying runtime

(b) varying λ

(c) varying b

(d) varying N

Figure 3.7: Experimental Results

and uniform strategies. We conclude that OSG is a better model against opportunistic criminals even with an inaccurate estimation of λ.

The COPS strategy, the SSG, and the uniform random strategy are compared again in Fig. 3.7(c), this time against criminals with different levels of anchoring bias b. In order to evaluate the performance of COPS when the defender has an inaccurate estimate of the anchoring bias b, we plotted both the expected utility of COPS where the defender has an accurate estimate of the criminal's anchoring bias and that using a fixed anchoring bias $b = 0.5$. $b = 0.5$ was picked from a set of sampled b since the OSG strategy with this b performs best. In Fig. 3.7(c), λ is fixed to 1, but experiments with other λ generate similar results. Again, COPS outperforms uniform random and SSG strategies.

To show COPS's scalability, we compare its performance with different numbers of defenders in metro systems with a varying number of stations; Five different settings are compared in Fig. 3.7(d): one defender, two defenders with full length patrolling, three defenders with full length patrolling, two defenders with segment patrolling, and three defenders with segment patrolling. The max runtime is 1800 s. With the same patrol techniques, more defenders provide higher expected utility. But, with the same amount of resources, segment patrolling outperforms full length patrolling.

Chapter 4

Learning OSG from Real Data

In this chapter, I introduce a model to recommend *optimal* police patrolling strategy against opportunistic criminals. I first build a game-theoretic model that captures the interaction between officers and opportunistic criminals. However, while different models of adversary behavior have been proposed, their exact form remains uncertain. Rather than simply hypothesizing a model as done in previous work, one key contribution is to learn the model from real-world criminal activity data. To that end, we represent the criminal behavior and the interaction with the patrol officers as parameters of a Dynamic Bayesian Network (DBN), enabling application of standard algorithms such as EM to learn the parameters.

4.1 Motivating Example

Domain Description: The motivating example for this study is the problem of controlling crime on a university campus. Our case study is about USC in USA. USC has a Department of Public Safety (DPS) that conducts regular patrols, similar to police patrols in urban settings. As part of our collaboration with USC DPS, we have access to

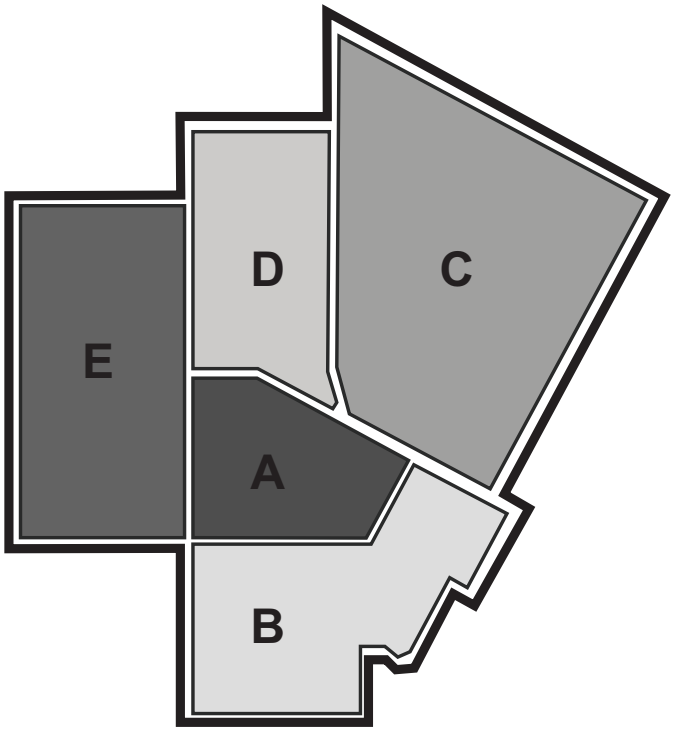


Figure 4.1: Campus map

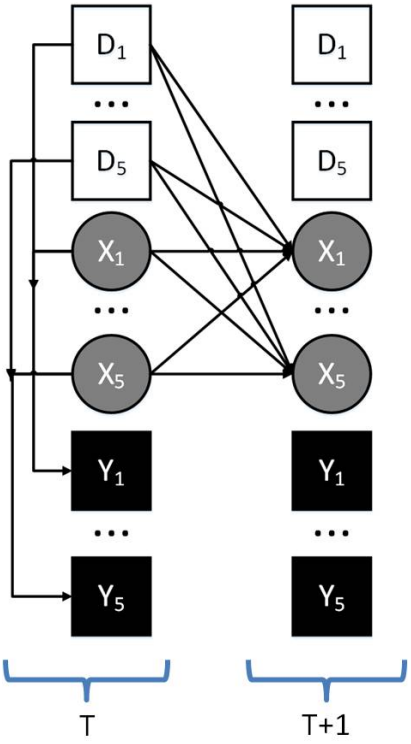


Figure 4.2: DBN for games

the crime report as well as patrol schedule on campus for the last three years (2011-2013). USC is a large enough university that allows us to claim that our methods are applicable to other large campuses, including large mall areas.

In USC, the campus map is divided into five patrol areas, which is shown in Fig 5.5. DPS patrols in three shifts per day. In the crime data all crimes are local, i.e., no crime happens across two patrol areas or patrol shifts. At the beginning of each patrol shift, DPS assigns each available patrol officer

Shift	A	B	C	D	E
1	1	1	2	2	2
2	1	1	1	2	1
3	2	1	1	3	1

Table 4.1: Crime data for 3 shifts.

Area	CaseNbr	ccClass	DateOccured	TimeOccured
D	1200668	DISTURBANCE	02/16/12	9:00
C	1200669	CHILD	02/16/12	10:08
B	1200672	TRAFFIC	02/16/12	11:23
C	1200674	TRAFFIC	02/16/12	15:25
A	1200675	THEFT-PETTY	02/16/12	15:10
C	1200676	SERVICE	02/16/12	15:20
D	1200677	PROPERTY	02/16/12	18:30
C	1200679	DOMESTIC	02/16/12	17:30
A	1200680	THEFT-PETTY	02/16/12	19:15

Figure 4.3: Sample Crime Report

to a patrol area and the officer patrols this area in this shift. At the same time, the criminal is seeking for crime opportunities by deciding which target they want to visit. Discussions with DPS reveal that criminals act opportunistically, i.e., crime is not planned in detail, but occurs when opportunity arise and there is insufficient presence of DPS officers.

AREA	DAY		
A	P3	*1060*	Oosterhof
A	P23	1062	Hudson
B	P22	*1051*	Bouligny
C	P51	1187	Ramirrez
D	P30	*1067*	Guerra
E	P46	*1061*	Harris

Figure 4.4: Patrol Schedule for 1 shift

There are two reports that DPS shared with us. The first is about criminal activity that includes details of each reported crime during the last three years, including the type of crime and the location and time information about the crime. We show a snapshot of this data in Figure 5.1. In my thesis, we do not distinguish between the different types

of crime and hence we consider only the number of crimes in each patrol area during each shift. Therefore, we summarize the three year crime report into $365 \times 3 \times 3 = 3285$ crime data points, one for each of the 8-hour patrol shift. Each crime data point contains five crime numbers, one for each patrol area.

The second data-set contains the DPS patrol allocation schedule. Every officer is allocated to patrolling within one patrol area. We show a snapshot of this data in Fig. 5.2. We assume that all patrol officers are homogeneous, i.e., each officer has the same effect on criminals' behavior. As a result, when generating a summary of officer patrol allocation data, we record only the number of officers allocated to each patrol area in each shift.

Table 5.2 shows a sample of the summarized officer patrol allocation data, where the row corresponds to a shift, the columns correspond to a patrol area and the numbers in each cell is the number of patrol officers. Table 5.1 shows a sample of the summarized crime data, where the row corresponds to a shift, the columns correspond to a patrol area and the numbers in each cell is the number of crimes. For example, from Table 5.2, we know that in shift 1, the number of officers in area A is 2 while the number of officers in area B, C, D and E is 1, while from Table 5.1 we know that in shift 1, there was 1 crime each in area A and B, and 2 crimes each in C, D and E. However, we do not know the number of criminals in any patrol area in any patrol shift. We call the patrol area as targets, and each patrol shift a time-step.

Problem Statement: Given data such as the real world data from USC, our goal is to build a general learning and planning framework that can

Shift	A	B	C	D	E
1	2	1	1	1	1
2	1	1	2	2	2
3	2	1	1	3	1

Table 4.2: Patrol data for 3 shifts.

be used to design optimal defender patrol allocations in any comparable urban crime setting.

We model the learning problem as a DBN, and we describe the basic model and the EM algorithm in the next section. Then, we present a compact form of our model that leads to improved learning performance. After that, we present methods to find the optimal defender plan for the learnt model with frequent update of the criminal model.

4.2 Learning Model

As stated earlier, we propose two approaches to learn the interaction between criminals and defenders: MCM and DBNM. The first approach, MCM directly relates crime distribution to observed data while MCM simply uses the number of unobserved criminals.

4.2.1 Markov models (MCM)

The models presented can be divided into three sub-categories: (1) crime as a function of crime history, (2) crime as a function of defender allocation and (3) crime as a function of crime history and defender allocation jointly. One motivation for this classification is to figure out the correlation between previous-time crime and previous or current-time defenders in the targets and find out if the presence of patrol officers affects the pattern of crime or not. We discuss these modeling approaches in the following sub-sections.

Crime predicts crime: In the first model shown in Fig. 4.5, we investigate prediction of crime based on crime distribution at the previous time step at the same target. This correlation is suggested based on some ideas introduced in criminology literature [36].

The desired correlation can be defined with the following mathematical function for all targets n: $Y_{n,t+1} = f(Y_{n,t})$.

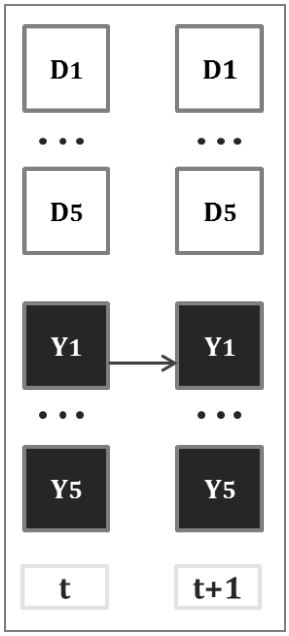


Figure 4.5: M1

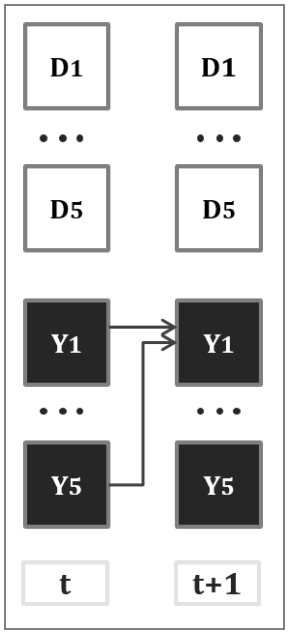


Figure 4.6: M2

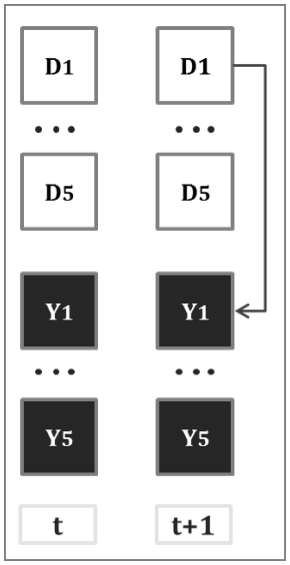


Figure 4.7: M3

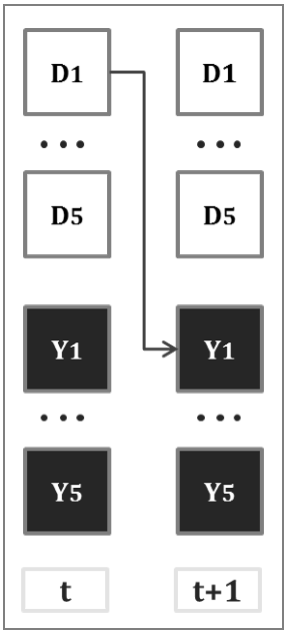


Figure 4.8: M4

To define and formalize the correlation and a pattern for crime prediction from history, we define a transition matrix, A, that represents how crime occurrences changes from one time step to the next one and apply maximum likelihood estimation to obtain it. In particular, $A(Y_{n,t}, Y_{n,t-1}) = P(Y_{n,t}|Y_{n,t-1})$.

For this model, probability for a sequence of events, i.e., Y_n which refers to numbers of crimes over a sequence of time steps, can be calculated as follows:

$$P(Y_n; A) = P(Y_{n,t}, ..., Y_{n,0}; A) = \prod_{1 \leq t \leq T} P(Y_{n,t}|Y_{n,t-1}; A) = \prod_{1 \leq t \leq T} A(Y_{n,t}, Y_{n,t-1})$$

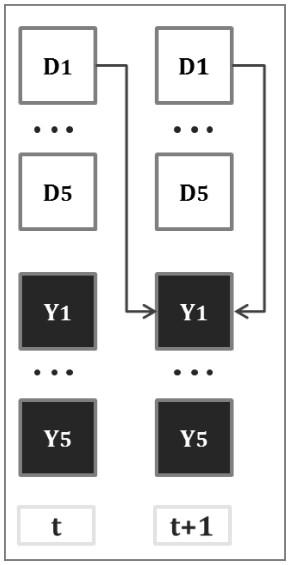


Figure 4.9: M5

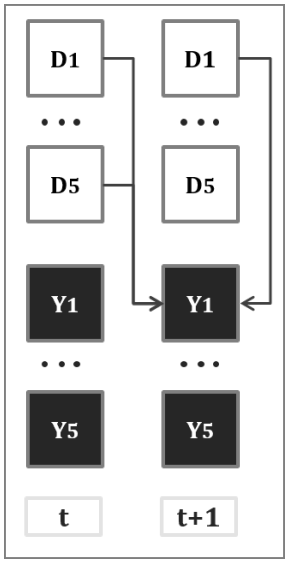


Figure 4.10: M6

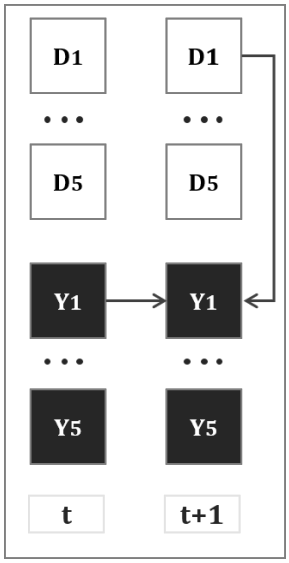


Figure 4.11: M7

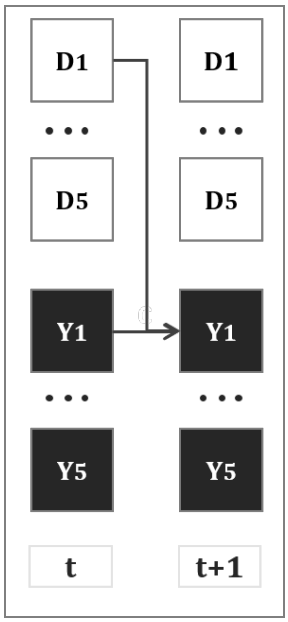


Figure 4.12: M8

In the above equations n indicates target number and A is the parameter to be estimated. Log likelihood for the above model for target i can be written as the following:

$$l(A) = \log P(Y_n; A) = \sum_{i=1}^{|S_Y|} \sum_{j=1}^{|S_Y|} \sum_{t=1}^{T} 1\{Y_{n,t} = S_i \wedge Y_{n,t-1} = S_j\} \log A_{ij}$$

where S_i and S_j indicates different values that Y can take and S_Y indicates the total possible number of values that Y can take, and 1 is the indicator function. As previously mentioned, in our case we made a binary assumption for variables, so they can take values zero and one. The optimization problem for maximizing the log-likelihood is :

$$\max_A l(A) \;\text{ subject to }\; \sum_{i=1}^{|S_Y|} A_{i,j} = 1 \text{ for } j = 1...|S_Y|,\; A_{ij} \geq 0 \text{ for } i, j = 1...|S_Y|.$$

The above optimization can be solved in closed form using the method of Lagrangian multipliers to obtain:

$$\hat{A_{ij}} = \frac{\sum_{t=1}^{T} 1\{Y_t = S_i \wedge Y_{t-1} = S_j\}}{\sum_{t=1}^{T} 1\{Y_{t-1} = S_j\}}$$

For each target we find a similar transition matrix. For all other models in this section, the same procedure for deriving the transition matrix is used. The model shown in Fig. 4.6 includes number of the crime at all other targets. This approach can be described with the following mathematical function: $Y_{n,t+1} = f(Y_{1:n,t})$.

Defender allocation predicts crime In the second approach we study the prediction of the crime based on the defender allocation. Four cases are studied: In Fig. 4.7, $Y_{n,t+1} = f(D_{n,t+1})$, which means the effect of the defender at the same target and time step is considered; in Fig. 4.8, $Y_{n,t+1} = f(D_{n,t})$, which means the defender allocation at previous step is considered; in Fig. 4.9, $Y_{n,t+1} = f(D_{n,t}, D_{n,t+1})$, which means the defender allocation in both the current and previous time step is considered; in Fig. 4.10, $Y_{n,t+1} = f(D_{1:n,t}, D_{n,t+1})$, meaning the defender allocation at all other targets from the previous time step and the defender allocation from the current time is considered. The same procedure as the previous subsection is used to find the transition matrix for the above models.

Crime and Defender allocation predicts crime In this sub-section we study the effect of crime and defender distribution jointly and investigate whether this combination improves the prediction. In Fig. 4.11, $Y_{n,t+1} = f(D_{n,t+1}, Y_{n,t})$, which means that the distribution of the crime at the previous step and the defender allocation at the current step is considered; in Fig. 4.12, $Y_{n,t+1} = f(D_{n,t}, Y_{n,t})$, this has a similar structure as the

previous one except that it considers the defender allocation at the previous time step; in Fig. 4.12, $Y_{n,t+1} = f(D_{n,t}, D_{n,t+1}, Y_{n,t})$, that is, the crime at that specific target is considered in addition to the defender allocation at the previous and current step.

4.2.2 Dynamic Bayesian Network Models (DBNM)

The second approach is based on the Dynamic Bayesian Network (DBN) model. A DBN is proposed in order to learn the criminals' behavior, i.e, how the criminals pick targets and how likely are they to commit crime at that target. This behavior is in part affected by the defenders' patrol allocation. In this section we assume that criminals are homogeneous, i.e., all criminals behave in the same manner.

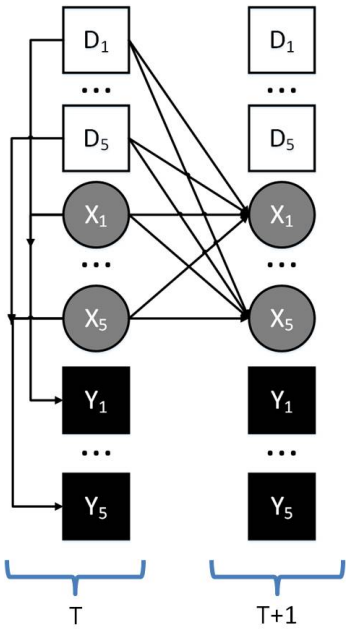


Figure 4.13: DBN for games

In every time-step of the DBN we capture the following actions: the defender assigns patrol officers to protect N patrol areas and criminals react to the defenders' allocation strategy by committing crimes opportunistically. Across time-steps the criminal can move from any target to any other, since a time-step is long enough to allow such a move. From a game-theoretic perspective, the criminals' payoff is influenced by the attractiveness of targets and the number of officers that are present. These payoffs drive the behavior of the criminals. However, rather than model the payoffs and potential bounded rationality of the criminals, we directly learn the criminal behavior as modeled in the DBN.

The DBN is shown in Fig 5.4: squares are observed states, where N white squares represent input states (number of defenders at each target) and N black squares represent output states (number of crime at each target) while N circles (number of criminals at each target) are hidden states. For ease of exposition, we use C to denote the largest value that any state can take. Next, we introduce the various parameters of this DBN.

DBN Parameters First, we introduce parameters that measure size of the problem

- N: Total number of targets in the graph.
- T: Total time steps of the training data.

Next, we introduce random variables for the observed state (input defender distribution and output crime distribution in our case) and the hidden state. We use three random variables to represent the global state for defenders, criminals and crimes at all targets.

- d_t: Defender's allocation strategy at step t: number of defenders at each target in step t with C^N possible values.
- x_t: Criminals' distribution at step t with C^N possible values
- y_t: Crime distribution at step t with C^N possible values.

Next, we introduce the unknown parameters that we wish to learn.

- π: Initial criminal distribution: probability distribution of x_1.
- A (*movement matrix*): The matrix that decides how x_t evolves over time. Formally, $A(d_t, x_t, x_{t+1}) = P(x_{t+1}|d_t, x_t)$. Given the C^N values for each argument of A, representing A requires $C^N \times C^N \times C^N$ parameters.

- B (*crime matrix*): The matrix that decides how criminals commit crime. Formally, $B(d_t, x_t, y_t) = P(y_t|d_t, x_t)$. Given the C^N values for each argument of B, representing B requires $C^N \times C^N \times C^N$ parameters.

Next, we introduce variables that are used in the EM algorithm itself. These variables stand for specific probabilities as illustrated below. We use d_i^j (y_i^j) as shorthand for $d_i, \ldots, d_j$ ($y_i, \ldots, y_j$):

- Forward prob.: $\alpha(k,t) = P(y_1^t, x_t = k|d_1^t)$
- Backward prob.: $\beta(k,t) = P(y_{t+1}^T|x_t = k, d_{t+1}^T)$
- Total prob.: γ: $\gamma(k,t) = P(x_t = k|y_1^T, d_1^T)$
- 2-step prob.: $\xi(k,l,t) = P(x_t = k, x_{t+1} = l|y_1^T, d_1^T)$.

The transition and observation probabilities in this model are multi-dimensional (> 2), i.e., they are tensors, but we call them matrix throughout the thesis. As shown in Fig 5.4, *movement matrix* A represents how the criminal distribution and DPS patrol strategy at step t affects the criminal distribution at step $t+1$; *crime matrix* B represents how the criminal distribution and DPS patrol strategy at step t affects the crime distribution at this step. Observe that we do not explicitly model the attractiveness of a target, as that factor is implicitly taken into account in A and B.

We can apply the EM algorithm to learn the unknown initial criminal distribution π, movement matrix A and output matrix B. However, EM applied to the basic DBN model above results in practical problems that we discuss in the next section.

Expectation Maximization EM is a class of algorithms for finding maximum likelihood estimation for unknown parameters in DBN [21]. The EM algorithm has an

initialization step, expectation (E) step and maximization (M) step. The initialization step chooses initial estimates for unknown parameters (π, A, B). The E step computes α, β, γ, ξ using these estimates. The M step updates the estimates of π, A, B using values of α, β, γ, ξ from E step. By iteratively performing E and M step, the EM algorithm converges to a local maxima of the likelihood function for parameters in the DBN. The particular mathematical equations used in E and M depends on the underlying model [9].

Initialization step

In our problem scenario, the EM algorithm is used to learn π, A and B from the given data for the observed states. The initial estimates of the variables should satisfy the condition

$$\sum_i \hat{\pi}(i) = 1, \sum_{x_{t+1}} \hat{A}(d_t, x_t, x_{t+1}) = 1, \sum_{y_t} \hat{B}(d_t, x_t, y_t) = 1 \ .$$

As EM only converges to local optima, we employ the standard method of running the algorithm with several randomly chosen initial condition in our experiments.

Expectation step

In the expectation step, we calculate the α, β, γ and ξ based on the current estimate of π, A and B, given by $\hat{\pi}, \hat{A}$ and $\hat{B}$

As is standard in inference in DBNs, $\alpha(k, 1)$ is calculated in a recursive manner. Hence, we first calculate forward probability at step 1, $\alpha(k, 1)$, which is shown in Eqn. 1

$$\alpha(k,1) = P(y_1, x_1 = k|d_1) = \hat{B}(d_1, k, Y_1) \cdot \hat{\pi}(k) \tag{4.1}$$

$$\alpha(k,t) = P(y_1, y_2, ..., y_t, x_t = k|d_1, d_2, ..., d_t)$$

$$= \sum_{x_{t-1}} \alpha(x_{t-1}, t-1)\hat{A}(d_{t-1}, x_{t-1}, x_t)\hat{B}(d_t, k, y_t) \tag{4.2}$$

$$\beta(k,T) = 1 \tag{4.3}$$

$$\beta(k,t) = P(y_{t+1}, y_{t+2}, ..., y_T|x_t = k, d_t, d_{t+1}, ..., d_T) =$$

$$\sum_{x_{t+1}} \beta(x_{t+1}, t+1)\hat{B}(d_{t+1}, x_{t+1}, y_{t+1})\hat{A}(d_t, k, x_{t+1}) \tag{4.4}$$

Given forward probability and backward probability at each step, the total probability γ is computed as shown in Eqn. 5 and the two step probability ξ is computed as shown in Eqn. 6.

$$\gamma(k,t) = P(x_t = k|y, d) = \frac{\alpha(k,t) \cdot \beta(k,t)}{\sum_k \alpha(k,t) \cdot \beta(k,t)} \tag{4.5}$$

$$\xi(k,l,t) = P(x_t = k, x_{t+1} = l|y, d)$$

$$= \frac{\alpha(k,t) \cdot A(d_t, k, l) \cdot \beta(l, t+1) \cdot B(d_{t+1}, l, y_{t+1})}{\sum_k \alpha(k,t) \cdot \beta(k,t)} \tag{4.6}$$

Maximization step

In maximization step, the estimate of π, A and B is updated using the probabilities we derive in expectation step.

$$\hat{\pi}(k) = \gamma(k, 1) \tag{4.7}$$

$$\hat{A}(d, k, l) = \frac{\sum_t 1_{d_t=d} \xi(k, l, t)}{\sum_t \sum_l 1_{d_t=d} \xi(k, l, t)} \tag{4.8}$$

$$\hat{B}(d, x, y) = \frac{\sum_t 1_{y_t=y, d_t=d} \gamma(x, t)}{\sum_t 1_{d_t=d} \gamma(x, t)} \tag{4.9}$$

where $1_{d_t=d}$ is an indicator function: $1_{d_t=d} = 1$ when $d_t = d$ and 0 otherwise. $1_{y_t=y,d_t=d}$ is also defined similarly. As a result, the new estimate of $A(d, k, l)$ is the ratio of the expected number of transitions from k to l given defender vector d to the expected total number of transitions away from k given defender vector d. The updated estimate of $B(d, x, y)$ is the ratio of the expected number of times the output of crimes equals to y while the defender is d, criminal is x to the total number of situations where the defender is d and criminal is x. To find a local optimal solution, the E and M steps is repeated, until $\hat{\pi}, \hat{A}, \hat{B}$ do not change significantly any more.

In EM algorithm, the size of movement matrix A is $C^N \times C^N \times C^N$ and the size of crime matrix B is also $C^N \times C^N \times C^N$. The number of unknown variables is $O(C^{3N})$. The exponentially many parameters make the model complex, and hence results in over-fitting given limited data. In addition, the time complexity as well as the space complexity of EM depends on the number of parameters, hence the problem scales exponentially with N. In practice, we can reduce C by categorizing the number of defenders, criminals and crimes. For example, we can partition the number of defenders, criminals and crimes into two categories each: the number of officers at each station is 1 (meaning ≤ 1) or 2 (meaning ≥ 2); the number of criminals/crimes is 0 (no criminal/ crime) or 1 (≥ 1

criminal/crime). However, the number of unknown parameters is still exponential in N. As a concrete example, in USC, $N = 5$ and the number of unknown parameters are more than 32768, even when we set $C = 2$. As we have daily data for three years, which is $365 \times 3 \times 3 = 3285$ data points, the number of parameters is much more than the number of data points. Therefore, we aim to reduce the number of parameters to avoid over-fitting and accelerate the computing process.

4.3 EM on Compact model (EMC^2)

In this section, we introduce our second contribution, which is to modify the basic DBN model to reduce the number of parameters. In the resultant compact model, the EM learning process runs faster and avoids over-fitting to the given data. The improvement may be attributed to the well-established learning principle of Occam's Razor [11], and our experimental results support our claims.

4.3.1 Compact model

We use three modifications to make our model compact. (1) We infer from the available crime data that crimes are local, i.e., crime at a particular target depends only on the criminals present at that target. Using this inference, we constructed a factored crime matrix B that eliminates parameters that capture non-local crimes. (2) Next, we rely on intuition from the Boyen-Koller [12] (BK) algorithm to decompose the joint distribution of criminals over all targets into a product of independent distributions for each target. (3) Finally, our consultations with the DPS in USC and prior literature on criminology [51] led us to conclude that opportunistic criminals by and large work independently. Using

this independence of behavior of each criminal (which is made precise in Lemma 4.3.1), we reduce the size of the movement matrix. After these steps, the number of parameters is only $O(N \cdot C^3)$.

Before describing these modifications in details, we introduce some notations that aid in describing the different quantities at each target: $Y_t = [Y_{1,t}, Y_{2,t}, ..., Y_{N,t}]$ is a N by 1 random vector indicating the number of crimes $Y_{i,t}$ at each target i at step t. D_t is a N by 1 random vector indicating the number of defenders $D_{i,t}$ at each target i at step t. X_t is a N by 1 random vector indicating the number of criminals $X_{i,t}$ at each target i at step t.

Factored crime matrix: The number of crime at one target at one step is only dependent on the criminals and officers present at that target at that step. Therefore, we factor the crime matrix B to a matrix that has an additional dimension with N possible values, to represent how the criminals and officers at one target decide the crime at that target. Therefore, instead of the original crime matrix B of size $C^N \times C^N \times C^N$ matrix, we have a factored crime matrix of size $N \times C \times C \times C$ crime matrix. The first dimension of factored crime matrix represents the target, the second dimension represents the number of defenders at this target, the third dimension represents the number of criminals and the fourth dimension represents the number of crimes. We still refer to this factored crime matrix as B, where $B(i, D_{i,t}, X_{i,t}, Y_{i,t}) = P(Y_{i,t}|D_{i,t}, X_{i,t})$

Marginalized hidden state: The BK algorithm presents an approximation method by keeping the marginals of the distribution over hidden states, instead of the full joint distribution. Following the BK intuition, we marginalize the hidden state, i.e., instead of considering the full joint probability of criminals at all targets (with C^N possible values),

we consider a factored joint probability that is a product of marginal probability of the number of criminals at each target.

In the unmodified DBN, the distribution over all the states at step t, $P(x_t)$ is a C^N by 1 vector. Additionally, the size of movement matrix A, which is the transition matrix from all the input and hidden state combinations at current step to the state at next step, is $C^N \times C^N \times C^N$. After marginalization, the marginals for each target i in the hidden state is $P(X_i = k, t)$, is a vector of size C. After we marginalize the hidden states, we only need to keep N marginals at each step, i.e., consider only N parameters. At each step, we can recover the distribution of full state by multiplying the marginals at this step. Then, we get the marginals at next step by evolving the recovered joint distribution of state at current step. Therefore, A can be expressed as a $C^N \times C^N \times N \times C$ matrix, where $A(d_t, x_t, i, X_{i,t+1}) = P(X_{i,t+1}|d_t, x_t)$.

Pairwise movement matrix A_m: Even with marginalized hidden state, we still need to recover the distribution of full state in order to propagate to next step. Therefore, the movement matrix size is still exponential with $C^N \times C^N \times N \times C$. In order to further reduce the number of unknown parameters and accelerate the computing process, we use properties of opportunistic criminals. Based on the crime reports and our discussion with DPS in USC, unlike organized terrorist attacks, the crimes on campus are committed by individual opportunistic criminals who only observe the number of defenders at the target they are currently at and do not communicate with each other. Therefore, at current step, the criminals at each target independently decide the next target to go to, based on their target-specific observation of number of defenders.

Based on the above observation, we can decompose the probability $P(X_{i,t+1} = 0|D_t, X_t)$ into a product of probabilities per target m. Denote by $X_{t+1}^{m\to i}$ the random variable that counts the number of criminals moving from target m to target i in the transition from time t to $t+1$. Lemma 4.3.1 proves that we can represent $P(X_{i,t+1} = 0|D_t, X_t)$ as a product of probabilities $P(X_{t+1}^{m\to i} = 0)$ for each m. $P(X_{t+1}^{m\to i} = 0)$ is a function of $D_{m,t}, X_{m,t}$

Lemma 4.3.1. *(**Independence of behavior**) For a N target learning problem, given the number of defenders at each location $D_t = [D_{1,t}, ..., D_{N,t}]$ and the number of criminals $X_t = [X_{1,t}, ..., X_{N,t}]$, the probability $P(X_{i,t+1} = 0|D_t, X_t)$ of the number of criminal being 0 at location i at step $t+1$ is given by $\prod_{j=1}^{N} P(X_{t+1}^{j\to i} = 0)$.*

Proof. Note that we must have $X_{t+1}^{m\to i} \geq 0$. We have the total number of criminals at target i at time step $t+1$ as $X_{i,t+1} = \sum_m X_{t+1}^{m\to i}$, i.e, the number of criminals at target i at step $t+1$ is the sum of criminals that move from each target to target i. Clearly $X_{i,t+1} = 0$ iff $X_{i,t+1}^{D_{m,t}, X_{m,t}} = 0$. Therefore, we have $P(X_{i,t+1} = 0|D_t, X_t) = P(X_{t+1}^{1\to i} = 0, \ldots, X_{t+1}^{N\to i} = 0)$. Since the criminals' decisions at each target are independent, we have $P(X_{t+1}^{1\to i} = 0, ..., X_{t+1}^{N\to i} = 0) = \prod_{m=1}^{N} P(X_{t+1}^{m\to i} = 0)$. □

When $C = 2$ and $X_{i,t} \in \{1, 2\}$, we can construct the whole movement matrix A using $P(X_{t+1}^{m\to i} = 0)$ (pairwise transition probabilities) by utilizing the fact that $P(X_{i,t+1} = 1|D_t, X_t) = 1 - P(X_{i,t+1} = 0|D_t, X_t)$. Therefore, instead of keeping A, we keep a transition matrix A_m where $A_m(i, D_{i,t}, X_{i,t}, j, X_{j,t+1}) = P(X_{t+1}^{i\to j})$. The number of parameters in A_m is $N \times 2 \times 2 \times N = 4N^2$. We do not consider the range of $X_{j,t+1}$ because we only need one parameter to store the two cases of $X_{j,t+1} = 1$ and $X_{j,t+1} = 0$ since

$A_m(i, D_{i,t}, X_{i,t}, j, X_{j,t+1} = 1) = 1 - A_m(i, D_{i,t}, X_{i,t}, j, X_{j,t+1} = 0)$. When $C > 2$, the number of variables in A_m are $C^2(C-1)N^2$; we can readily extend the above construction of A from A_m.

One simple example is when $X_{i,t} \in \{0, 1, 2\}$, we can apply Lemma 1 to derive

$$\begin{aligned}
P(X_{i,t+1} = 0|D_t, X_t) &= \textstyle\prod_{j=1}^{N} P(X_{i,t+1} = 0|D_{j,t}, X_{j,t}) \\
P(X_{i,t+1} = 1|D_t, X_t) &= \textstyle\sum_{j=1}^{N} [P(X_{i,t+1} = 1|D_{j,t}, X_{j,t}) \\
&\quad \textstyle\prod_{k=1,k\neq j}^{N} P(X_{i,t+1} = 0|D_{k,t}, X_{k,t})] \\
P(X_{i,t+1} = 2|D_t, X_t) &= 1 - P(X_{i,t+1} = 0|D_t, X_t) \\
&\quad -P(X_{i,t+1} = 1|D_t, X_t)
\end{aligned}$$

4.3.2 EMC^2 procedure

EM on CompaCt model (EMC^2) procedure applies the EM algorithm to the compact DBN model. To learn the initial distribution $\pi_{k,i} = P(X_{i,1} = k)$, matrix A_m and matrix B, we first generate initial estimates of these parameters that satisfy the condition $\sum_k \hat{\pi}(k, i) = 1$, $\sum_{X_{j,t+1}} \hat{A}_m(i, D_{i,t}, X_{i,t}, j, X_{j,t+1}) = 1$ and
$\sum_{Y_{i,t}} \hat{B}(i, D_{i,t}, X_{i,t}, Y_{i,t}) = 1$.

Next, we define the intermediate variables used in the EM algorithm. These differ from the earlier application of EM because of our changed model. We use the shorthand Y_i^j to denote $Y_i, ..., Y_j$ and D_i^j to denote $D_i, ..., D_j$:

Forward prob.: $\alpha(i, k, t) = P(Y_1^t, X_{i,t} = k|D_1^t)$,

Backward prob.: $\beta(i, k, t) = P(Y_{t+1}^T|X_{i,t} = k, D_t^T)$,

Total prob.: $\gamma(i,k,t) = P(Y, X_{i,t} = k|D_1^T)$,

2-step prob.: $\xi(i,k,j,l,t) = P(X_{i,t} = k, X_{j,t+1} = l|Y_1^T, D_1^T)$.

Next, the E and M steps are used with random restarts to learn the values of π, A_m and B. While the equations used in the E and M steps can be derived following standard EM techniques, we illustrate a novel application of the distributive law for multiplication in the E step that enables us to go from exponential time complexity to polynomial (in N) time complexity. Without going into details of the algebra in the E step, we just focus on the part of the E step that requires computing $P(Y_1^{t-1}, X_{i,t} = 0|D_1^t)$.

The following can be written from total law of probability

$$P(Y_1^{t-1}, X_{i,t} = 0|D_1^t) = \sum_{X_{t-1}} P(Y_1^{t-1}, X_{i,t} = 0, X_{t-1}|D_1^t)$$
$$= \sum_{X_{t-1}} P(Y_1^{t-1}|D_1^t, X_{i,t} = 0, X_{t-1}) P(X_{i,t} = 0|D_1^t, X_{t-1})$$
$$P(X_{t-1}|D_1^t)$$

The above can be simplified using the Markovian assumptions of the DBN to the following

$$\sum_{X_{t-1}} P(Y_1^{t-1}|D_1^t, X_{t-1}) P(X_{i,t} = 0|D_{t-1}, X_{t-1}) P(X_{t-1}|D_1^t)$$

The first and third term can be combined (Bayes theorem) to obtain

$$\sum_{X_{t-1}} P(Y_1^{t-1}, X_{t-1}|D_1^t) P(X_{i,t} = 0|D_{t-1}, X_{t-1})$$

Using the Boyen-Koller assumption in our compact model we get

$$P(Y_1^{t-1}, X_{t-1}|D_1^t) = \prod_j P(Y_1^{t-1}, X_{j,t-1}|D_1^t)$$

Also, using Lemma 4.3.1 we get

$$P(X_{i,t} = 0|D_{t-1}, X_{t-1}) = \prod_j P(X_t^{j\to i} = 0)$$

Thus, using these we can claim that $P(Y_1^{t-1}, X_{i,t} = 0|\mathcal{D}_t)$ is

$$\sum_{X_{t-1}} \prod_j P(Y_1^{t-1}, X_{j,t-1}|D_1^t) P(X_t^{j\to i} = 0)$$

Since the range of X_{t-1} is C^N, naively computing the above involves summing C^N terms, thus, implying a time complexity of $O(C^N)$. The main observation that enables polynomial time complexity is that we can apply principles of the generalized distributive law [1] to reduce the computation above. As an example, the three summations and four multiplication in $ab + ac + bc + bd$ can be reduced to two summations and one multiplication by expressing it as $(a + b)(c + d)$. Using distributive law we reduce the computation for $P(Y_1^{t-1}, X_{i,t} = 0|D_1^t)$ by switching sum and product

$$\prod_j \sum_{X_{j,t-1}} P(Y_1^{t-1}, X_{j,t-1}|D_1^t) P(X_t^{j\to i} = 0)$$

The complexity of computing the above is $O(N^C)$. Applying this idea, we can calculate α, β, γ and ξ from the estimated value of $\hat{\pi}$, $\hat{A_m}$ and $\hat{B}$ in the expectation step in time polynomial in N.

For the maximization step, we update the estimate of π, A_m and B using the probabilities we derive in the expectation step. The procedure is the same as Equation (7) to (9).

4.3.3 Variable definitions

Y_t is a N by 1 vector indicating the appearance of crime at each target at step t; D_t is a N by 1 vector indicating the appearance of defender at each target at step t; X_t is a N by 1 vector indicating the appearance of criminal at each target at step t. $Y_{i,t}$ indicates the appearance of crime at target i at step t; $D_{i,t}$ indicates the appearance of defender at target i at step t; $Y_{i,t}$ indicates the appearance of criminal at target i at step t.

Example

$Y_t = [Y_{1,t}, Y_{2,t}]$, *where* Y_t *is a vector with* $N = 2$ *elements* $Y_{i,t}$*; so is* $D_t, D_{i,t}$ *and* $X_t, X_{i,t}$

For simplicity, we set $X_{i,t}$ and $Y_{i,t}$ as binary variables with value 0 and 1, indicating there is no criminal/crime at target i at step t and there is at least 1 criminal/crime at target i at step t. Also, we set $D_{i,t}$ as a variable with 2 values: $D_{i,t} = 1$ means there is at most 1 defender at target i at step t, $D_{i,t} = 2$ means there is at least 2 defender at target i at step t. These variables can be easily extended to those with more than 2 values.

Criminal Distribution π: π is a 2 by T by N matrix. The three dimensions are the appearance probability of criminals at target i at step t, inference of step t, inference of target i.

Example

Element $\pi(k, t, i)$ represents the probability of appearance of k criminal(s) at target i at step t, e.g. $\pi(0, 1, 1)$ is the probability that at step 1, there is 0 criminal at target 1. Based on our binary setting, we do have $\pi(0, 1, 1) + \pi(1, 1, 1) = 1$, which is also true for other stations and steps

Transition Matrix A: A is a 2^N by 2^N by 2 by N matrix. The four dimensions are the appearance of defender at current step d_t, the appearance of criminal at current step x_t, the appearance of criminal at next step at target i, the inference of i.

Example

We encode D_t(1 by N vector) with a number d_t (2^N possible values). e.g. $d_t = 1$ equals to $[D_{1,t} = 1, D_{2,t} = 1]$; $d_t = 2$ equals to $[D_{1,t} = 2, D_{2,t} = 1]$; $d_t = 3$ equals to $[D_{1,t} = 1, D_{2,t} = 2]$; $d_t = 4$ equals to $[D_{1,t} = 2, D_{2,t} = 2]$; we use the similar way to encode X_t with x_t.

Element $A(d_t, x_t, x_{i,t+1}, i)$ represents given d_t and x_t at step t, the probability of criminal's appearance $x_{i,t+1}$ at station i at $t+1$, e.g. $A(2, 1, 0, 2) = 0.1$ means that if at step t, $D_{1,t} = 2$(2 defenders at target 1), $D_{1,t} = 1$ and $X_{1,t} = 0, X_{2,t} = 0$, there is 0.1 probability that at step $t + 1$ there are no criminals at target 2.

Output Matrix B: B is a 2 by 2 by 2 by N matrix. The four dimensions are the appearance of crime at target i, the appearance of criminal at target i, the appearance of defender at target i, the inference of i.

Example

Element $B(y, x, d, i)$ represents given the criminal's appearance x and the defender's appearance d at target i, the probability of crime's appearance is y. $B(1, 1, 2, 2) = 0.2$ means if there are 1 criminal and 2 defender at target 2, there is 0.2 probability that there will be a crime.

Forward probability α: α is a 2 by T by N matrix. The three dimensions are the appearance of criminals at target i at step t, inference of step t, inference of target i.

Element $\alpha(k, t, i) = P(Y_1, ..., Y_t, X_{i,t} = k | D_1, ..., D_t)$

Backward probability β: β is a 2 by T by N matrix. The three dimensions are the appearance of criminals at target i at step t, inference of step t, inference of target i.

Element $\beta(k, t, i) = P(Y_{t+1}, ..., Y_T | X_{i,t} = k, D_t, ..., D_T)$

Total probability γ: γ is a 2 by T by N matrix. The three dimensions are the appearance of criminals at target i at step t, inference of step t, inference of target i.

Element $\gamma(k, t, i) = P(X_{i,t} = k | Y, D)$

Two-step probability ξ: ξ is a 2 by T by N by 2^N matrix. The four dimensions are the appearance of criminals at target i at step $t+1$, inference of step t, inference of target i, the appearance of criminals at step t.

Element $\xi(k,t,i,x_t) = P(X_t = x_t, X_{i,t+1} = k|Y,D)$

4.3.4 Initialization step

Set random initial conditions for start criminal distribution $\pi(k,1,i)$, Transition Matrix A and Output Matrix B

Example

The initial distribution of criminals $\pi(k,1,i)$, transition matrix A and output matrix B are all set to be uniform distribution, which is shown below.

$\pi(0,1,1) = 0.5, \pi(1,1,1) = 0.5;$

$\pi(0,1,2) = 0.5, \pi(1,1,2) = 0.5;$

$A(1,1,0,1) = 0.5, A(1,1,1,1) = 0.5; A(1,1,0,2) = 0.5, A(1,1,1,2) = 0.5;$

...

$A(4,4,0,1) = 0.5, A(4,4,1,1) = 0.5; A(4,4,0,2) = 0.5, A(4,4,1,2) = 0.5;$

$B(0,0,1,1) = 0.5, B(1,0,1,1) = 0.5;$

...

$B(0,1,2,2) = 0.5, B(1,1,2,2) = 0.5;$

This initialization means that at time step 1, for each target there are 0.5 probability there is a criminal and 0.5 probability that there is no criminal. Also, the probability for criminal to transit from any target i at step t to any target j at step $t+1$ is 0.5. Finally, the probability of criminal committing a crime at any target is 0.5 with or without defender. After iteratively executing expectation and maximization step, the initial variables will converges to the real distribution.

4.3.5 Expectation step

The main idea of the expectation step is to calculate the forward probability α and backward probability β based on the knowledge of estimation of $\pi(k, 1, i)$, Transition matrix A and Output matrix B.

Step 1: iteratively calculating forward probability

Firstly calculate forward probability at step 1 $\alpha(k, 1, i)$

$$\alpha(k, 1, i) = P(Y_{i1}, X_{i1} = k|D_{i1}) = B(y, k, d, i) \cdot \pi(k, 1, i) \tag{4.10}$$

The equation holds because $B(y, k, d, i) = P(Y_{i1}|X_{i1} = k, D_{i1})$ and $\pi(k, 1, i) = P(X_{i1} = k) = P(X_{i1} = k|D_{i1})$: At the beginning $P(X_{i1} = k)$ is given and independent from D_{i1}

Then iteratively calculate forward probability from step 2 to T $\alpha(k, t, i)$

$$\alpha(k,t,i) = P(Y_1, Y_2, ..., Y_t, X_{it} = k | D_1, D_2, ..., D_t)$$

$$= \sum_{X_{t-1}} [P(X_{i,t} = k | X_{t-1}) \cdot P(Y_1, Y_2, ..., Y_{t-1}, X_{t-1} | D_1, D_2, ..., D_{t-1})]$$

$$\cdot P(Y_{it} | X_{it} = k, D_{it})$$

$$= \sum_{X_{t-1}} [A(D_{t-1}, X_{t-1}, k, i) \cdot \prod_{j=1}^{N} \alpha(X_{j,t-1}, t-1, j)] \cdot B(y, k, d, i) \tag{4.11}$$

We play with the conditional probability here.

Step 2: iteratively calculating backward probability

Firstly calculate backward probability at step T $\beta(k,T,i)$

$$\beta(k,T,i) = 1 \tag{4.12}$$

Then iteratively calculate forward probability from step T-1 to 1 $\beta(k,t,i)$

$$\beta(k,t,i) = P(Y_{t+1}, Y_{t+2}, ..., Y_T | X_{i,t} = k, D_t, D_{t+1}, ..., D_T)$$

$$= \sum_{X_t : X_{i,t} = k \in X_t} [\sum_{X_{t+1}} P(Y_{t+2}, Y_{t+3}, ..., Y_T | X_{t+1}, D_{t+1}, .., D_T)$$

$$\cdot P(Y_{t+1} | X_{t+1}, D_{t+1}) P(X_{t+1} | X_t, D_t)]$$

$$= \sum_{X_{t+1}} \sum_{X_t : X_{i,t} = k \in X_t} \prod_{j=1}^{N} [\beta(X_{j,t+1}, t+1, j)$$

$$\cdot B(Y_{j,t+1}, X_{j,t+1}, D_{j,t+1}, j) \cdot A(D_t, X_t, X_{j,t+1}, j)] \tag{4.13}$$

This step is similar to calculating forward probability

4.3.6 Maximization step

The main idea of the maximization step is to update the new estimation of $\pi(k, 1, i)$, Transition matrix A and Output matrix B based on new-calculated forward/backward probability α and β.

Step 1: Calculating total probability γ

$$\begin{aligned}
\gamma(k,t,i) &= P(X_{i,t} = k|Y, D) \\
&= P(Y|X_{i,t} = k, D) \cdot \frac{P(X_{i,t} = k, D)}{P(Y, D)} \\
&= P(Y_1, ..., Y_t|X_{i,t} = k, D) \cdot P(Y_{t+1}, ..., Y_T|X_{i,t} = k, D) \cdot \frac{P(X_{i,t} = k, D)}{P(Y, D)} \\
&= P(Y_1, ..., Y_t|X_{i,t} = k, D) \cdot P(Y_{t+1}, ..., Y_T, X_{i,t} = k|D) \cdot \frac{P(D)}{P(Y, D)} \\
&= \frac{\alpha(k,t,i) \cdot \beta(k,t,i)}{\sum_k \alpha(k,t,i) \cdot \beta(k,t,i)}
\end{aligned} \tag{4.14}$$

Still we play with the transformation of conditional probability. There are two important points in this equation: 1, From second line to the third line, we use the conditional independence of $Y_1, ..., Y_t$ and $Y_{t+1}, ..., Y_T$ given $X_{i,t} = k, D$; 2, in the last line, we use the normalization to represents $\frac{P(D)}{P(Y,D)}$

Step 2: Calculating two-step probability ξ

$$\xi(k,t,i,x_t) = P(X_t = x_t, X_{i,t+1} = k|Y,D)$$

$$= P(Y|X_t = x_t, X_{i,t+1} = k, D) \cdot P(X_{i,t+1} = k|X_t = x_t, D) \cdot P(X_t = x_t, D) \cdot \frac{1}{P(Y,D)}$$

$$= P(Y_1,..,Y_t|X_t = x_t, X_{i,t+1} = k, D) \cdot P(Y_{t+1}|X_t = x_t, X_{i,t+1} = k, D)$$

$$\cdot P(Y_{t+2},...,Y_T|X_t = x_t, X_{i,t+1} = k, D) \cdot P(X_{i,t+1} = k|X_t = x_t, D)$$

$$\cdot P(X_t = x_t, D)\frac{1}{P(Y,D)} = P(Y_1,..,Y_t|X_t = x_t, D_1,..,D_t) \cdot P(Y_{t+1}|X_{i,t+1} = k, D_{t+1})$$

$$\cdot P(Y_{t+2},...,Y_T|X_{i,t+1} = k, D_{t+1},...,D_T) \cdot P(X_{i,t+1} = k|X_t = x_t, D)$$

$$\cdot P(X_t = x_t, D)\frac{1}{P(Y,D)} = P(Y_1,..,Y_t, X_t = x_t, D_1,..,D_t) \cdot P(Y_{t+1}|X_{i,t+1} = k, D_{t+1})$$

$$\cdot P(Y_{t+2},...,Y_T|X_{i,t+1} = k, D_{t+1},...,D_T) \cdot P(X_{i,t+1} = k|X_t = x_t, D) \cdot \frac{1}{P(Y,D)}$$

$$= \frac{\prod_j \alpha(x_{j,t},t,j) \cdot A(D_t,x_t,k,i) \cdot \beta_k(t+1) \cdot B(Y_{i,t+1},k,D_{i,t+1},i)}{\sum_k \sum_{x_t} \prod_j \alpha(x_{j,t},t,j) \cdot A(D_t,x_t,k,i) \cdot \beta_k(t+1) \cdot B(Y_{i,t+1},k,D_{i,t+1},i)} \tag{4.15}$$

This calculation is very complex. However, the idea is similar to the way we calculate γ. From second line to third line, we use the conditional independence of $Y_1,..,Y_t$ and Y_{t+1} and $Y_{t+2},...,Y_T$ given $X_t, X_{i,t+1}, D$. Again, we use the normalization to represent $\frac{1}{P(Y,D)}$

Step 3: Update the initial state distribution:

$$\pi(k,1,i) = \gamma(k,1,i) \tag{4.16}$$

by definition, $\pi(k,1,i)$ is γ at step 1, which is the expected frequency of value k for target i at time 1

Step 4: Update Output matrix B

$$B(y,x,d,i) = \frac{\sum_t 1_{Y_{it}=y,D_{it}=d}\gamma(x,t,i)}{\sum_t 1_{D_{it}=d}\gamma(x,t,i)} \tag{4.17}$$

$$1_{Y_{it}=y,D_{it}=d} = \begin{cases} 1, Y_{it} = y, D_{it} = d \\ 0, otherwise \end{cases}$$

$$1_{D_{it}=d} = \begin{cases} 1, D_{it} = d \\ 0, otherwise \end{cases}$$

where $1_{Y_{it}=y,D_{it}=d}$ *is an indicator function and and* $B(y,x,d,i)$ *is the expected number of times the output of target* i *crimes have been equal to* y *while the defender is* d*, criminal is* k *over the expected total number of times.*

Step 5: Update Transition Matrix A

$$A(d,\theta,k,i) = \frac{\sum_t 1_{D_t=d}\xi(k,t,i,\theta)}{\sum_t \sum_j 1_{D_t=d}\xi(j,t,i,\theta)} \tag{4.18}$$

which is the expected number of transitions from criminal vector θ *to criminal number* k *at target* i *given defender vector* d *compared to the expected total number of transitions away from criminal vector* θ *given defender vector* d*.*

4.3.7 Repeat

Repeat expectation step and maximization step, which is $\pi, A, B, \gamma, \xi \rightarrow \alpha, \beta \rightarrow \pi, A, B, \gamma, \xi$, until π, A, B do not change anymore. This means we reach the local optimal solution.

EMC^2 procedure

4.3.8 Initialization step

Set random initial conditions for start criminal distribution π, Transition Matrix A and Output Matrix B

Example

For initial step, we just randomly set the number. A simple way is using the uniform distribution, which is:(of course random starts is an better option...)

$\pi(0,1) = 0.5, \pi(1,1) = 0.5, \pi(0,2) = 0.5, \pi(1,2) = 0.5;$

$A(1,1,0,1,0) = 0.5, A(1,1,0,1,1) = 0.5, A(1,1,0,2,0) = 0.5, A(1,1,0,2,1) = 0.5;$

...

$A(2,2,1,1,0) = 0.5, A(2,2,1,1,1) = 0.5, A(2,2,1,2,0) = 0.5, A(2,2,1,2,1) = 0.5;$

$B(1,1,1,0) = 0.5, B(1,1,1,1) = 0.5, B(1,1,2,0) = 0.5, B(1,1,2,1) = 0.5;$

...

$B(2,2,0,0) = 0.5, B(2,2,0,1) = 0.5, B(2,2,1,0) = 0.5, B(2,2,1,1) = 0.5;$

4.3.9 Expectation step

The main idea of the expectation step is to calculate the forward probability α and backward probability β based on the estimation of π, A and B.

Step 1: iteratively calculating forward probability

Firstly calculate forward probability at step 1 $\alpha(i, k, 1)$

$$\alpha(i,k,1) = P(Y_1, X_{i,1} = k|D_1) = P(Y_{i,1}, X_{i,1} = k|D_{i,1})$$

$$= P(Y_{i,1}|X_{i,1} = k, D_{i,1}) \cdot P(X_{i,1} = k|D_{i,1}) = B(i, D_{i,1}, k, Y_{i,1}) \cdot \pi(k, i)$$

Then iteratively calculate forward probability from step 2 to T $\alpha(i, k, t)$. We apply dynamic programming to go one step forward. We use $P(Y_1, Y_2, ..., Y_{t-1}, X_{i,t}|D_1, D_2, ..., D_{t-1})$ as an intermediate variable.

First, we calculate $P(Y_1, Y_2, ..., Y_{t-1}, X_{i,t}|D_1, D_2, ..., D_{t-1})$ using $\alpha(m, k, t-1)$:

$$P(Y_1, Y_2, ..., Y_{t-1}, X_{i,t}|D_1, D_2, ..., D_{t-1})$$

$$= \sum_{X_{t-1}} P(Y_1, Y_2, ..., Y_{t-1}, X_{t-1}, X_{i,t}|D_1, D_2, ..., D_{t-1})$$

$$= \sum_{X_{t-1}} P(Y_1, Y_2, ..., Y_{t-1}|X_{i,t}, X_{t-1}, D_1, D_2, ..., D_{t-1}) P(X_{i,t}|X_{t-1}, D_{t-1}) P(X_{t-1})$$

$$= \sum_{X_{t-1}} P(Y_1, Y_2, ..., Y_{t-1}|X_{t-1}, D_1, D_2, ..., D_{t-1}) P(X_{i,t}|X_{t-1}, D_{t-1}) P(X_{t-1})$$

$$= \sum_{X_{t-1}} P(Y_1, Y_2, ..., Y_{t-1}, X_{t-1}, D_1, D_2, ..., D_{t-1}) P(X_{i,t}|X_{t-1}, D_{t-1})$$

$$if X_{i,t} = 0$$

$$= \sum_{X_{t-1}} \prod_m \alpha(m, X_{m,t-1}, t-1) \prod_m P(X_{i,t} = 0|X_{m,t-1}, D_{m,t-1})$$

$$= \prod_m \sum_{X_{m,t-1}} \alpha(m, X_{m,t-1}, t-1) P(X_{i,t} = 0|X_{m,t-1}, D_{m,t-1})$$

$$if X_{i,t} = 1$$

$$= \sum_{X_{t-1}} \prod_m \alpha(m, X_{m,t-1}, t-1)$$

$$[\prod_m \sum_{x_{i,t}} P(X_{i,t} = x_{i,t}|X_{m,t-1}, D_{m,t-1}) - \prod_m P(X_{i,t} = 0|X_{m,t-1}, D_{m,t-1})]$$

$$= \prod_m \sum_{X_{m,t-1}} \alpha(m, X_{m,t-1}, t-1) \sum_{x_{i,t}} P(X_{i,t} = x_{i,t}|X_{m,t-1}, D_{m,t-1})$$

$$- \prod_m \sum_{X_{m,t-1}} \alpha(m, X_{m,t-1}, t-1) P(X_{i,t} = 0|X_{m,t-1}, D_{m,t-1}) \quad (4.19)$$

The complexity is this step is $O(N^2)$ to calculate the whole intermediate vector. Next, we calculate $\alpha(i, k, t)$ using $P(Y_1, Y_2, ..., Y_{t-1}, X_{i,t}|D_1, D_2, ..., D_{t-1})$.

$$\alpha(i,k,t) = P(Y_1, Y_2, ..., Y_t, X_{i,t} = k | D_1, D_2, ..., D_t)$$

$$= \sum_{X_t : X_{i,t}=k} P(Y_1, Y_2, ..., Y_t, X_t | D_1, D_2, ..., D_t)$$

$$= \sum_{X_t : X_{i,t}=k} P(Y_1, Y_2, ..., Y_t | X_t, D_1, D_2, ..., D_t) P(X_t | D_1, D_2, ..., D_t)$$

$$= \sum_{X_t : X_{i,t}=k} P(Y_1, Y_2, ..., Y_{t-1} | X_t, D_1, D_2, ..., D_t)$$

$$P(Y_t | X_t, D_1, D_2, ..., D_t) P(X_t | D_1, D_2, ..., D_t)$$

$$= \sum_{X_t : X_{i,t}=k} P(Y_1, Y_2, ..., Y_{t-1} | X_t, D_1, D_2, ..., D_t)$$

$$P(Y_t | X_t, D_1, D_2, ..., D_t) P(X_t | D_1, D_2, ..., D_t)$$

$$= \sum_{X_t : X_{i,t}=k} P(Y_1, Y_2, ..., Y_{t-1}, X_t | D_1, D_2, ..., D_{t-1}) P(Y_t | X_t, D_t)$$

$$= \sum_{X_t : X_{i,t}=k} \prod_j P(Y_1, Y_2, ..., Y_{t-1}, X_{j,t} | D_1, D_2, ..., D_{t-1}) \prod_j P(Y_{j,t} | X_{j,t}, D_{j,t})$$

$$= \prod_{j \neq i} \sum_{X_{j,t}} P(Y_1, Y_2, ..., Y_{t-1}, X_{j,t} | D_1, D_2, ..., D_{t-1}) P(Y_{j,t} | X_{j,t}, D_{j,t})$$

$$\cdot P(Y_1, Y_2, ..., Y_{t-1}, X_{i,t} = k | D_1, D_2, ..., D_{t-1}) P(Y_{i,t} | X_{i,t} = k, D_{i,t}) \tag{4.20}$$

The complexity to calculate each element is $O(N)$ and the total complexity is still $O(N^2)$

Step 2: iteratively calculating backward probability

Firstly calculate backward probability at step T $\beta(i,k,T)$

$$\beta(i,k,T) = 1 \tag{4.21}$$

Then iteratively calculate forward probability from step T-1 to 1 $\beta(k,t,i)$

$$\begin{aligned}
\beta(i,k,t) &= P(Y_{t+1},...,Y_T|X_{i,t}=k,D_t,...,D_T) \\
&= \sum_{X_{t+1}} P(Y_{t+1},Y_{t+2},...,Y_T,X_{t+1}|X_{i,t},D_t,D_{t+1},...,D_T) \\
&= \sum_{X_{t+1}} P(Y_{t+1},Y_{t+2},...,Y_T|X_{t+1},X_{i,t},D_t,D_{t+1},...,D_T)P(X_{t+1}|X_{i,t},D_t) \\
&= \sum_{X_{t+1}} P(Y_{t+1},Y_{t+2},...,Y_T|X_{t+1},D_t,D_{t+1},...,D_T)P(X_{t+1}|X_{i,t},D_t) \\
&= \sum_{X_{t+1}} P(Y_{t+2},...,Y_T|X_{t+1},D_t,D_{t+1},...,D_T) \\
&P(Y_{t+1}|X_{t+1},D_t,D_{t+1},...,D_T)P(X_{t+1}|X_{i,t},D_t) \\
&= \sum_{X_{t+1}} \prod_m (P(Y_{t+2},...,Y_T|X_{m,t+1},D_t,D_{t+1},...,D_T) \\
&P(Y_{m,t+1}|X_{m,t+1},D_{m,t+1})P(X_{m,t+1}|X_{i,t},D_t)) \\
&= \prod_m \sum_{X_{m,t+1}} (P(Y_{t+2},...,Y_T|X_{m,t+1},D_t,D_{t+1},...,D_T) \\
&P(Y_{m,t+1}|X_{m,t+1},D_{m,t+1})P(X_{m,t+1}|X_{i,t},D_t)) \\
&= \prod_m \sum_{X_{m,t+1}} (\beta(m,X_{m,t+1},t+1) \\
&B(m,D_{m,t+1},X_{m,t+1},Y_{m,t+1})A(i,D_{i,t},X_{i,t},m,X_{m,t+1}))
\end{aligned} \tag{4.22}$$

This step is similar to calculating forward probability. The complexity is still $O(N)$

Step 3: Calculating total probability γ

$$
\begin{aligned}
\gamma(i,k,t) &= P(X_{i,t}=k|Y,D) \\
&= \sum_{X_t:X_{i,t}=k} P(X_t|Y,D) \\
&= \sum_{X_t:X_{i,t}=k} P(Y|X_t,D)\cdot\frac{P(X_t,D)}{P(Y,D)} \\
&= \sum_{X_t:X_{i,t}=k} P(Y_1,...,Y_t|X_t,D)\cdot P(Y_{t+1},...,Y_T|X_t,D)\cdot\frac{P(X_t,D)}{P(Y,D)} \\
&= \sum_{X_t:X_{i,t}=k} P(Y_1,...,Y_t|X_t,D)\cdot P(Y_{t+1},...,Y_T,X_t|D)\cdot\frac{P(D)}{P(Y,D)} \\
&= \alpha(i,k,t)\cdot\beta(i,k,t)\prod_{j\neq i}\sum_{X_{j,t}}\alpha(j,X_{j,t},t)\beta(j,X_{j,t},t)\cdot\frac{P(D)}{P(Y,D)} \\
&= \frac{\alpha(i,k,t)\cdot\beta(i,k,t)}{\sum_k \alpha(i,k,t)\cdot\beta(i,k,t)}
\end{aligned}
\tag{4.23}
$$

Still we play with the transformation of conditional probability. There are two important points in this equation: 1, From second line to the third line, we use the conditional independence of $Y_1,...,Y_t$ and $Y_{t+1},...,Y_T$ given $X_t=k,D$; 2, in the last line, we use the normalization

Step 4: Calculating two-step probability ξ

$$\xi(m, x_{m,t}, i, x_{i,t+1}, t) = P(X_{m,t} = x_{m,t}, X_{i,t+1} = x_{i,t+1}|Y, D)$$

$$= P(Y|x_{m,t}, x_{i,t+1}, D) \cdot P(x_{i,t+1}|x_{m,t}, D) \cdot P(x_{m,t}, D)$$

$$= P(Y_1, .., Y_t|x_{m,t}, x_{i,t+1}, D) \cdot P(Y_{t+1}, ..., Y_T|x_{m,t}, x_{i,t+1}, D) \cdot P(x_{m,t}, D)P(x_{i,t+1}|x_{m,t})$$

$$= P(Y_1, .., Y_t, x_{m,t}, D) \cdot P(Y_{t+1}, ..., Y_T|x_{i,t+1}, D)P(x_{i,t+1}|x_{m,t}) = P(Y_1, .., Y_t, x_{m,t}, D)$$

$$\cdot \sum_{X_{t+1}:X_{i,t+1}=x_{i,t+1}} P(Y_{t+2}, ..., Y_T|X_{t+1}, D)P(Y_{t+1}|X_{t+1}, D)P(x_{i,t+1}|x_{m,t})$$

$$= P(Y_1, .., Y_t, x_{m,t}, D) \cdot P(X_{i,t+1}|X_{m,t})P(Y_{t+2}, ..., Y_T|X_{i,t+1}, D)P(Y_{i,t+1}|X_{i,t+1}, D)$$

$$\prod_{j\neq i} \sum_{X_{j,t+1}} P(Y_{t+2}, ..., Y_T|X_{j,t+1}, D)P(Y_{j,t+1}|X_{j,t+1}, D)P(X_{j,t+1}|X_{m,t})$$

$$= \alpha(m, x_{m,t}, t) \cdot A(i, D_{i,t}, X_{i,t}, m, X_{m,t+1})\beta(j, X_{i,t+1}, t+1)$$

$$\prod_{j\neq i} \sum_{X_{j,t+1}} (\beta(j, X_{j,t+1}, t+1)B(j, D_{j,t+1}, X_{j,t+1}, Y_{j,t+1})A(i, D_{i,t}, X_{i,t}, j, X_{j,t+1})) \quad (4.24)$$

This calculation is similar to the way we calculate γ. From second line to third line, we use the conditional independence of $Y_1, .., Y_t$ and Y_{t+1} and $Y_{t+2}, ..., Y_T$ given X_t, X_{t+1}, D. Again, we use the normalization to represent $\frac{1}{P(Y,D)}$. Edit: maybe we can simplify the calculation. However, this is not important since it is $O(N)$

4.3.10 Maximization step

The main idea of the maximization step is to update the new estimation of π, Transition matrix A and Output matrix B based on new-calculated forward/backward probability α and β.

Step 1: Update the initial state distribution:

$$\pi(k,i) = \gamma(i,k,1) \tag{4.25}$$

by definition, π is γ at step 1, which is the expected frequency of value k at time 1

Step 2: Update Output matrix B

$$B(i,d_{i,t},x_{i,t},y_{i,t}) = \frac{\sum_t 1_{Y_{i,t}=y_{i,t},D_{i,t}=d_{i,t}}\gamma(i,x_{i,t},t)}{\sum_t 1_{D_{i,t}=d_{i,t}}\gamma(i,x_{i,t},t)} \tag{4.26}$$

$$1_{Y_{i,t}=y_{i,t},D_{i,t}=d_{i,t}} = \begin{cases} 1, Y_{i,t}=y_{i,t}, D_{i,t}=d_{i,t} \\ 0, otherwise \end{cases}$$

$$1_{D_{i,t}=d_{i,t}} = \begin{cases} 1, D_{i,t}=d_{i,t} \\ 0, otherwise \end{cases}$$

where $1_{Y_{i,t}=y_{i,t},D_{i,t}=d_{i,t}}$ is an indicator function and $B(i,d,x,y)$ is the expected number of times the output of crimes have been equal to y while the defender is d, criminal is k over the expected total number of times at target i.

Step 3: Update Transition Matrix A

$$A(m,d_{m,t},x_{m,t},i,x_{i,t+1}) = \frac{\sum_t 1_{D_t=d_t}\xi(m,d_{m,t},x_{m,t},i,x_{i,t+1}))}{\sum_t \sum_{x_{i,t+1}} 1_{D_t=d_t}\xi(m,d_{m,t},x_{m,t},i,x_{i,t+1}))} \tag{4.27}$$

which is the expected number of transitions from x_t to k given defender vector d_t compared to the expected total number of transitions away from x_t given defender vector d_t.

4.3.11 Repeat

Repeat expectation step and maximization step, which is $\pi, A, B \to \alpha, \beta, \gamma, \xi \to \pi, A, B$, until π, A, B do not change anymore. This means we reach the local optimal solution.

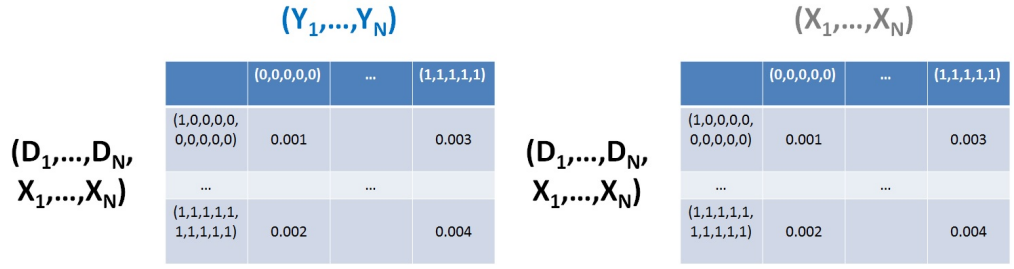
Computational complexity analysis The complexity of EM on the basic model is $O(C^{2N}T)$, and for EMC^2 it is $O(N^{C+1}T + (C \cdot N)^2 T)$. The detailed derivation is not hard and delegated to the appendix. Therefore, EMC^2 procedure runs much faster than EM in the basic model when C is small.

4.4 Example of EMC^2

In this section, I want to give an example of how EMC^2 compactly represents the parameters in the DBN model.

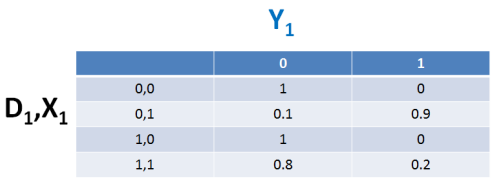
4.4.1 Factorize crime matrix

In the original DBN model shows in Figure 5.4, the crime matrix is shown in Figure 4.14(a). The size of this crime matrix is 2^{2N} by 2^N for game with N targets. This is because the input entry of crime matrix is the combination of defenders' allocation at N target (2^N) and criminals' possible distribution at N targets (2^N) and the output entry is the crime distribution at N targets (2^N). In EMC^2 model, we compactly represents such transition by assuming crime at each targets are independent. An example of the

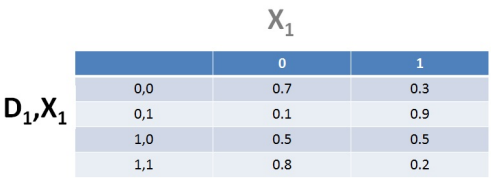


(a) Original Crime Matrix

(b) Original Transition Matrix



(c) Factorized Crime Matrix



(d) Pairwise Transition Matirx

Figure 4.14: Example of learnt parameters

factorized crime matrix for target 1 is show in Figure 4.14(c). For all other targets, the size of the crime matrix is the same. Therefore, the total size of the factorized crime matrix is $4by2byN$, which is significantly smaller.

4.4.2 Pairwise transition matrix

Similarly, in the original DBN model, the output matrix is shown in Figure 4.14(b). The size of transition matrix is also 2^{2N} by 2^N for game with N targets. The input entry of transition matrix is the combination of defenders' allocation at N target at time step T (2^N) and criminals' possible distribution at N targets (2^N) at time step T. The output entry is the criminals' distribution at N targets (2^N) at time step $T+1$. In EMC^2 model, we assume the transition matrix can be reconstructed by using the pairwised transition matrix from each target i to each target j. An example of the pairwised transition matrix from target 1 to target 1 is shown in Figure 4.14(d). For all other target pairs, the size

of the transition matrix is the same. Therefore, the total size of the pairwised transition matrix is $4by2byN^2$, which is significantly also smaller.

4.5 Dynamic Planning

The next step after learning the criminals' behavior is to design effective officer allocation strategies against such criminals. In this section, we first introduce a simple online planning mechanism, in which we iteratively update criminals' behavior model and plan allocation strategies. Next, we present a slower optimal planning algorithm and faster but sub-optimal greedy algorithm.

Online Planning Mechanism. We first state our template for iterative learning and planning before describing the planning algorithms. The criminal behavior may change when the criminal observes and figures out that the defender strategy has changed. Thus, the optimal strategy planned using the learned parameters is no longer optimal after some time of deployment of this strategy, as the parameters itself change in response to the deployed strategy.

To address the problem above, we propose an online planning mechanism. In this mechanism, we update criminal's model based on real-time crime/patrol data and dynamically plan our allocation strategy. The first step is to use the initial training set to learn an initial model. Next, we use a planning algorithm to generate a strategy for the next T_u steps. After executing this strategy, we can collect more crime data and use them to update the model with the original training data. By iteratively doing this, we

generate strategies for the whole horizon of T steps. Algorithm 3 presents the details of this mechanism.

Input: $Train\_data$: the training crime and patrol data; T_u: the time steps to plan ahead in one iteration; T: the total time steps to plan ahead
Output: $[D_1, ..., D_T]$: The patrol strategies for T times.
$A, B, \pi \leftarrow Learn(Train\_data)$ $t = 0$ **while** $t < T$ **do**
  $[D_t, ..., D_{t+T_u}] \leftarrow Plan(A, B, \pi)$ $[Y_1, ..., Y_{T_u}] \leftarrow Execute\{D_1, ..., D_{T_u}\}$
  $Train\_data \leftarrow Train\_data \cup \{D_1, Y_1, ..., D_{T_u}, Y_{T_u}\}$
  $A, B, \pi \leftarrow Update(Train\_data, A, B, \pi)$ $t = t + T_u$;
end

Algorithm 3: Online planning

Compared to simply applying planning algorithm for T steps, our online planning mechanism updates criminals' behavior model periodically based on his response to the currently deployed strategy. In this online planning mechanism, three parts are needed: learning algorithm, updating algorithm and planning algorithm. For learning and updating algorithm, we apply the EMC^2 learning algorithm from Section 5. In addition, we also need a planning algorithm, which we discuss next.

4.5.1 Planning Algorithms

The planning problem. In the planning problem, the criminals' behavior is known, or more specifically, we already know the criminals' initial distribution π, movement matrix A and crime matrix B in the DBN model. Given a pure defender patrol allocation strategy for T_u steps, we can plug those values for the input state in the DBN and get the expected number of crimes in T_u steps. The goal of planning is to find the defenders' pure strategy that optimizes the defenders' utility, which in our case is to minimize the total expected number of crimes. (In our way our framing, any randomized strategy, which is

the combination of pure strategies, results in more number of crimes than the optimal pure strategy). Thus, planning against opportunistic criminals is a search problem in defender's pure strategy space. First, we present the practical impossibility of a brute force search.

Brute Force search: A naive way to solve this problem is to try all possible allocation strategies and pick the one that leads to least crimes in T_u steps. However, since at each step, the number of possible allocation strategies is C^N and there are T_u steps in total, the strategy space is C^{NT_u}. For example, for our specific problem of patrolling in USC with five targets, two categories and the goal of planning for $T_u = 300$ steps, we need to search $2^{1500} \approx 10^{451}$ different strategies, which is impractical to solve.

Input: A: Criminal's transition matrix; B: Crime matrix; π: Criminal's initial distribution
Output: $\widehat{D}$: Defender's optimal patrol strategy.

for *each officer allocation* D_1^i **do**
  $\mathsf{Parent}[i,1] \leftarrow 0$; $P_{i,1} \leftarrow f_Y(A, \pi, D_1^i)$; $X_{i,1} \leftarrow \pi$ **for** $t \leftarrow 2, 3, ..., T_u$ **do**
    for *each officer allocation* D_t^j **do**
      Let $F(i) = f_Y(f_X(A, X_{i,t-1}, D_{t-1}^i), D_t^j, B) + P_{i,t-1}$
      $\mathsf{Parent}[D_t^j, t] \leftarrow \text{argmin}_i[F(i)]$ $P_{j,t} \leftarrow \min_i[F(i)]$
      $X_{j,t} \leftarrow f_X(A, X_{\mathsf{Parent}[D_t^j,t],t-1}, D_{t-1}^{\mathsf{Parent}[D_t^j,t]})$
    end
    $index[T] \leftarrow \text{argmin}_i P_{i,T}$; $\widehat{D}[T] \leftarrow D_T^{index[T]}$
  end
  for $t \leftarrow T-1, ..., 1$ **do**
    $index[t] \leftarrow \mathsf{Parent}[D_t^{index[t+1]}, t+1]$ $\widehat{D}[t] \leftarrow D_t^{index[t]}$
  end
end

Algorithm 4: DOGS

Dynamic Opportunistic Game Search (DOGS): First, we list some notation that will be used in the next two planning algorithms.

- D_t^j indicates the j^{th} strategy for the defender from the C^N different defender strategies at time step t.
- $P_{j,t}$ is the total number of crimes corresponding to the optimal defender strategy for the first t time-steps that has j as its final defender strategy.
- $X_{j,t}$ is the criminals' location distribution corresponding to the optimal defender strategy for the first t time-steps that has j as its final defender strategy.
- $f_Y(X_t, D, B)$ is the expected number of crimes at all targets at t given the criminal location distribution X_t and defender's allocation strategy D at step t and output matrix B.
- $f_X(A, X_t, D_t)$ is the criminal location distribution at step $t+1$ given the criminal location distribution X_t and defender's allocation strategy D_t at t and transition matrix A.

DOGS is a dynamic programming algorithm, hence in order to find the optimal strategy for t steps, we first find the optimal strategy for the sub-problem with $t-1$ steps and use it to build the optimal strategy for t steps. Given the values of π, A and B from our learning step, the optimal defender allocation strategy $D_1, ..., D_{T_u}$ is given by the recurrence relations:

$$P_{j,1} = f_Y(\pi, D_1^j, B)$$
$$P_{j,t} = \min_i[f_Y(f_X(A, X_{i,t-1}, D_{t-1}^i), D_t^j, B) + P_{i,t-1}]$$

Retrieving the optimal allocation strategy requires remembering the allocation D^i_{t-1} that minimizes the second equation, which is done by storing that information in the function Pa, as follows:

$$\mathsf{Pa}[j,t] = \operatorname*{argmin}_i [f_Y(f_X(A, X_{i,t-1}, D^i_{t-1}), D^j_t, B) + P_{i,t-1}]$$

As P_{j,T_u} is the total number of crime for the optimal defender strategies for T_u time-steps that has j as the final strategy, the optimum strategy for time-step T_u is given by $D_{T_u} = \operatorname{argmin}_j P_{j,T_u}$. Then, recursively, given optimal D_t we find the optimal strategy in the previous time-step using function Pa: $D_{t-1} = \mathsf{Pa}[D_t, t]$. The complexity of DOGS algorithm (Algorithm 4) is $O(C^{2N}T_u)$.

Greedy search. The dynamic programming based algorithm can generate the optimal strategy, but takes time $O(C^{2N}T_u)$. We present a greedy algorithm that runs in $O(C^N T_u)$ time, but the solution may be sub-optimal. In greedy search, we split the strategy space into T_u slices. Each slice represents the strategy at each step. Then, instead of searching the optimal strategy for T_u steps, we only look one step ahead to search the strategy that optimize defender's utility at current step (Algorithm 5). It finds the optimal patrol allocation D_t at current step by minimizing the expected number of crime at all targets at step t. For the next step, we compute the criminal's distribution X_{t+1} and greedily search again. We keep iterating this process until we reach T_u step. The complexity of Greedy search is $O(C^N T_u)$.

Input: A: Criminal's transition matrix; B: Crime matrix; π: Criminal's initial distribution

Output:

$$D = [D_1, ..., D_{T_u}]$$

: Defender's optimal patrol strategy.

1 **for** $t \leftarrow 1, \ldots, T_u$ **do**

2 $\quad D_t \leftarrow \text{argmin}_D\, f_Y(X_t, D, B)$; $X_{t+1} \leftarrow f_X(A, X_t, D_t)$

3 **end**

Algorithm 5: GREEDY

4.6 Experimental Results

Experimental setup. All our experiments were performed on a machine with 2.4GHz and 16GB RAM. MATLAB was our choice of programming language. There are two threads of experiments, one on learning and other on learning and planning. To avoid leaking confidential information of USC Department of Public Safety, all the crime numbers shown in the results are normalized.

Learning (Setting): Our first experiment is on evaluating performance of EMC^2 algorithm in learning criminals' behavior. We use the case study of USC in our experiments. We obtained three years of crime report and corresponding patrol schedule followed in USC. Since EMC^2 algorithm and EM algorithm only reach locally optimal solution, we run the algorithms for 30 different randomly chosen start points and choose the best solution from among these runs. These start points, i.e., values of A, B and π, are generated by sampling values from a uniform random distribution over $[0, 1]$ for all the elements and then normalizing the probabilities so that they satisfy the initial conditions. C is set to 2 by default while the effect of varying C is compared in Figure 4.19.

Results:

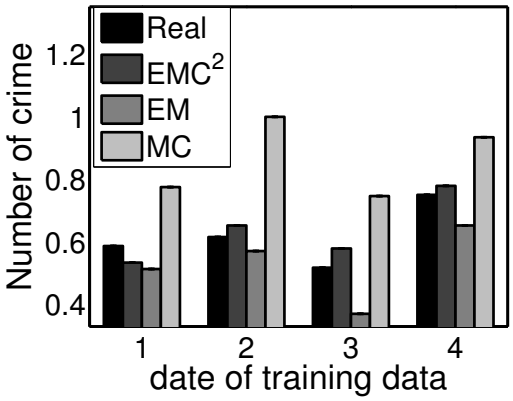


Figure 4.15: Total number of crime

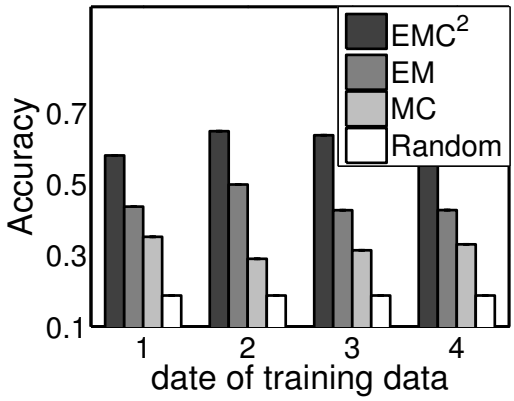


Figure 4.16: Individual Accuracy

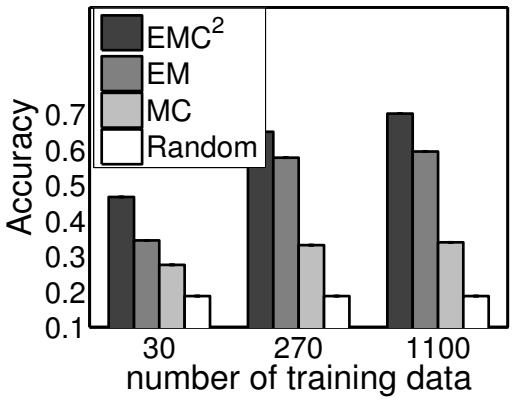


Figure 4.17: Varying data

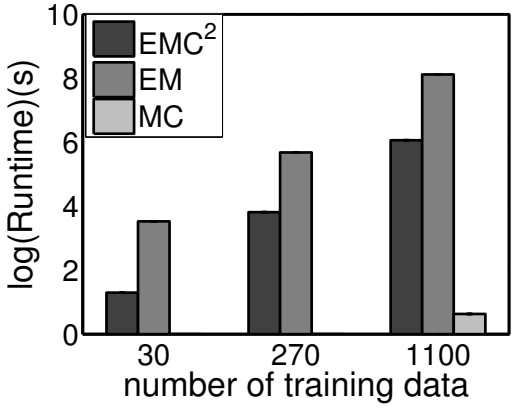


Figure 4.18: Varying data(Runtime)

The results shown in Figure 4.15 compares the estimated numbers of crimes using different learning algorithms with real number of crimes in 30 days. Three different algorithms are compared: (1) the Markov chain (MC) algorithm, in which the best performance among all eight models are shown, (2) the exact EM algorithm and (3) the EMC^2 algorithm. We divide the three year data into four equal parts of nine months each. For each part we train on the first eight months data and test on the ninth month data. The x-axis in this figure indicates the index of the part of data that we evaluate on. y-axis is the total number of crimes in 30 days. The closer this number is to the real number of crime, the better the prediction is. As can be seen, the prediction of EMC^2

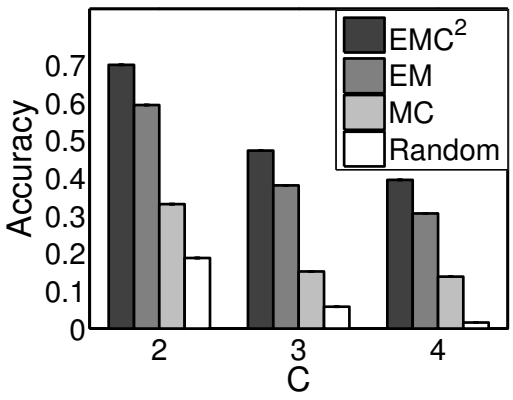


Figure 4.19: Vary C

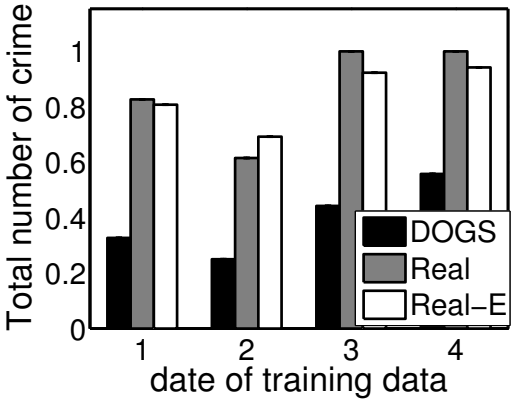


Figure 4.20: Compare with deployed

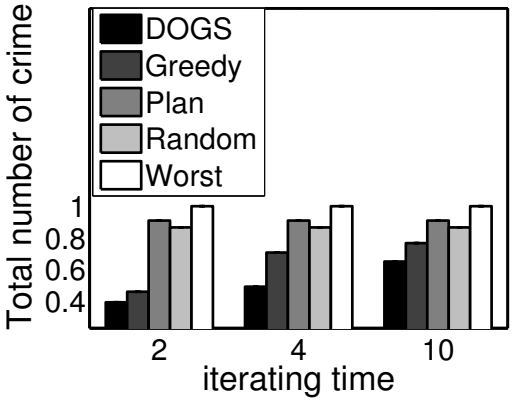


Figure 4.21: Vary T_u

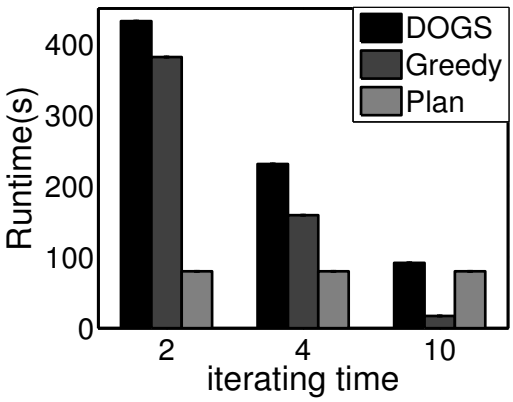


Figure 4.22: Vary T_u(Runtime)

is much closer compared to those of EM and MC algorithm in all the training groups. This indicates that the crime distribution is related to criminals' location and including number of criminals at each target as a hidden state helps improving performance. In addition, EMC^2 algorithm achieves better performance than EM by reducing number of unknown variables to avoid over-fitting.

For Figure 4.16, we measure learning performance for each individual target using a metric that we call accuracy. To define this metric, let n_{it} be the actual number of crimes at target i for time step t, let n'_{it} be the predicted number of crimes at target i at time step t. Then, accuracy at step t is the probability of the event $\sum_{i=1}^{N} |n_{it} - n'_{it}| \leq 1$. In other

words, it is the probability that we make less than one mistake in predicting crimes for all N targets. The reported accuracy is the average accuracy over all t. In Figure 4.16, the y-axis represents the accuracy. The higher accuracy is, the more accurate our prediction is. We compare four different algorithm: MC, EM, EMC^2 algorithm and the uniform random algorithm, which sets equal probability for all possible numbers of crimes at each target. As expected, EMC^2 outperforms all other algorithms in all training groups. In addition, even though the accuracy of the algorithms varies in different training groups, which we attribute to the noisy nature of the data in the field, the largest difference is within 15%. This indicates accuracy of the algorithms are data-independent.

We present additional results under this setting in Figure 4.17 and 4.18. We compare the four approaches for varying size of training data, thus, the x-axis in both figures shows the number of training data (in days of data) used in learning. Our test data is all of the data points from a 30 day period, and the training data are the data points just before (in order of time) the test data points. For Figure 4.17, EMC^2 algorithm again outperforms all other algorithms for any number of training data in accuracy. In addition, the more data we have for training, the better accuracy we achieve. In Figure 4.18, the y-axis shows runtime in seconds on a log scale. The more data we have, the longer it takes for each training method. Random algorithm is pre-generated and takes almost no time, hence that data is not shown in the figure; the runtime for MC is negligible because the number of state is small ($O(4^N)$) and we traverse all the data points only once; the runtime for EMC^2 algorithm is significantly better than that for EM algorithm, as is expected by our complexity analysis in Section 5.

In Figure 4.19, we compare the four approaches by varying C. The x-axis shows the value of C. We use 1100 data points for training while 30 data points, which is just after the training data points, are used for testing. The accuracy decreases as C increases. This is because when C increases, there are more possible values of number of crimes. Thus, the possibility of predicting an accurate number decreases. However, when C increases from 3 to 4, the decrease in accuracy is small in EMC^2 due to the fact that data with value 4 rarely appears in both the crime and patrol data-set. This indicates a small C is a good approximation. In addition, EMC^2 algorithm again outperforms all other algorithms for any C.

In Figure 4.23, all the Markov Chain models are compared. The x-axis is the model index. RP represents the uniform random predicting. $M1$ through $M9$ are the nine Markov Chain models. The y-axis represents the accuracy. We use four sets of data that are the same as Figure 4.16 to test the performance. All the Markov chain models give similar accuracy, which outperforms RP significantly. However, as showed in Figure 4.16, all these models are outperformed by EM and EMC^2 algorithms.

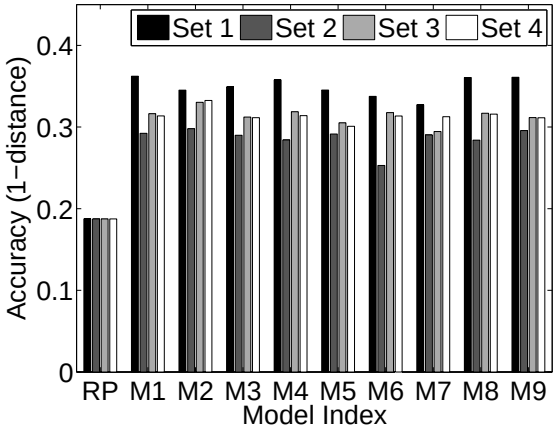


Figure 4.23: Markov Chain Models

Learning and Planning (Real world data): Figure 4.20 compares DOGS with the actual deployed allocation strategy generated by DPS experts in USC. Similar to the settings in Figure 4.15, we divide the three year data into four equal parts of nine months.

For each part we train on the first eight months data using EMC^2 algorithm and test different allocation strategy on the first 10 days of the ninth month data. When testing the strategy, we assume the criminals' behavior remain unchanged during these 10 days. Three different scenarios are compared: (1) the real number of crimes, shown as Real in Fig. 4.20; (2) the expected number of crimes with DPS strategy and learned criminal behavior, shown as Real-E and (3) the expected numbers of crime with DOGS allocation and learned criminal behavior, shown as DOGS. As shown in Fig 4.20, the expected number of crime with DPS strategy is close to the real number of crimes, which indicates EMC^2 captures the main features of the criminal behavior and provides close estimate of the number of crimes. In addition, DOGS algorithm outperforms the strategy generated by domain experts significantly. This demonstrates the effectiveness of DOGS algorithm as compared to current patrol strategy. By using allocation strategy generated by DOGS, the total crime number reduces by $\sim$50% as compared to the currently deployed strategy.

Learning and planning (Simulated data): Next, we evaluate the performance of our online planning mechanism. We use simulations for this evaluation. In the simulation, the criminal model is simulated using the model from an earlier work on opportunistic criminals [59], in which the authors explicitly model an opportunistic criminal's behavior. However, the defender does not know the type of criminals in our experiments. Instead, the defender starts by executing a random patrol schedule for 300 steps and collects the corresponding crime report using which they learn an initial criminal behavior model. The criminal responds to the defenders' patrol schedule as predicted by the behavior model in [59]. Since the criminal behavior in [59] is probabilistic, we run the experiment 30 times and each data point we report in this part is an average over these 30 instances.

We fix the number of patrol officers to $2N - 2$, where N is the number of targets. This number is consistent with our real data-set numbers (8 officers for 5 targets), where there were enough officers to allocate one officer to each target, but not enough to allocate two officers to each target. We use EMC^2 algorithm as the learning algorithm.

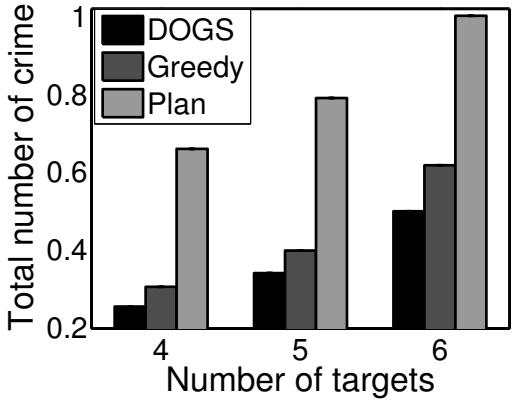


Figure 4.24: Varying N

Results: Figure 4.21 to 4.24 presents the results from our experiments about the online learning and planning mechanism. Four planning mechanisms that we consider are as follows: first, a random planning mechanism that randomly generates allocation strategy with limited resources; second, a pure planning mechanism, where we learn the criminal behavior model once and apply this model to plan for the entire horizon T using DOGS algorithm; third, a online planning mechanism with greedy planning algorithm that updates every T_u time-steps; and the last mechanism is online planning mechanism with DOGS algorithm that also updates every T_u time-steps. In Figure 4.21, the total planning horizon T is set to 600. In addition to the four planning mechanisms, we also consider the worst case where the defender always protect the least valuable targets. The x-axis shows the update interval T_u, which is the time interval after which we update criminals' behavior model. The y-axis is the expected number of crimes that happens under the deployed allocation strategy within 600 steps. Expected number of crimes under pure planning mechanism stay the same with different T_u because it does not update the criminals' model at all.

For online mechanisms, the expected number of crimes increases as the update interval T_u increases. This is because with infrequent updates of the criminals' behavior model, we cannot keep up with the real criminals' behavior. In addition, with any size of the update interval, DOGS algorithm outperforms the greedy algorithm. In Figure 4.22, we present the runtime of three mechanisms for the same experiment. We do not show the runtime for the random planning mechanism as it is small and same for any planning horizon T. The runtime decreases as the update interval T_u increases. There is a runtime-quality trade-off in choosing T_u. Figure 4.24 shows the performance of the four planning mechanisms, but with different number of targets in the model. The x-axis is the number of targets in the graph and the y-axis is the expected number of crimes under the deployed strategy. We set $T = 600$, $T_u = 2$. The results here are similar to the results of Fig. 4.21.

These results lead us to conclude that online mechanisms outperform the baseline planning mechanisms significantly in any settings. For online mechanisms, DOGS achieves better performance while greedy planning algorithm requires less runtime. Thus, based on the specific problem being solved, the appropriate algorithm must be chosen judiciously.

Chapter 5

Large Scale OSG

In this chapter, I aim to deter urban crime by recommending optimal police patrol strategies against *opportunistic criminals* in large scale urban problems. While previous work has tried to learn criminals' behavior from real world data and generate patrol strategies against opportunistic crimes, it cannot scale up to large-scale urban problems. Our first contribution is a game abstraction framework that can handle opportunistic crimes in large-scale urban areas. In this game abstraction framework, we model the interaction between officers and opportunistic criminals as a game with discrete targets. By merging similar targets, we obtain an abstract game with fewer total targets. We use real world data to learn and plan against opportunistic criminals in this abstract game, and then propagate the results of this abstract game back to the original game. Our second contribution is the layer-generating algorithm used to merge targets as described in the framework above. This algorithm applies a mixed integer linear program (MILP) to merge similar and geographically neighboring targets in the large scale problem. As our third contribution, we propose a planning algorithm that recommends a mixed strategy against opportunistic criminals. Finally, our fourth contribution is a heuristic propagation model

to handle the problem of limited data we occasionally encounter in large-scale problems. As part of our collaboration with local police departments, we apply our model in two large scale urban problems: a university campus and a city. Our approach provides high likelihood in the real datasets; furthermore, we project significant crime rate reduction using our planning strategy compared to current police strategy.

5.1 Problem Statement

Area	CaseNbr	ccClass	DateOccured	TimeOccured
D	1200668	DISTURBANCE	02/16/12	9:00
C	1200669	CHILD	02/16/12	10:08
B	1200672	TRAFFIC	02/16/12	11:23
C	1200674	TRAFFIC	02/16/12	15:25
A	1200675	THEFT-PETTY	02/16/12	15:10
C	1200676	SERVICE	02/16/12	15:20
D	1200677	PROPERTY	02/16/12	18:30
C	1200679	DOMESTIC	02/16/12	17:30
A	1200680	THEFT-PETTY	02/16/12	19:15

Figure 5.1: Sample Crime Report

AREA		DAY	
A	P3	*1060*	Oosterhof
A	P23	1062	Hudson
B	P22	*1051*	Bouligny
C	P51	1187	Ramirrez
D	P30	*1067*	Guerra
E	P46	*1061*	Harris

Figure 5.2: Sample Schedule

In my thesis, we focus on limiting opportunistic crimes in large scale urban areas. Such large scale areas are usually divided into N targets by the defenders. At the same time, defenders divide the time into patrol shifts. T denotes the total number of shifts. At the beginning of each patrol shift, the defender assigns each available patrol officer to a target and the officer patrols this target in this shift. The criminals observe the defender's allocation and seek crime opportunities by deciding the target to visit. In order to learn the criminal's opportunistic reaction to the defender's allocation, two categories of data are required for T shifts. The first is about crime activity which contain crime details. Figure 5.1 shows a snapshot of this kind of data in a campus region. In my thesis, we only consider the time and location information of crimes, ignoring the difference among

different types of crimes. Therefore, we can summarize the crime report into a table like Table 5.1. In this table, columns represents the index of each target while rows represents total number of shifts, 1...T. Each element in the table represents the number of crimes at the corresponding target in that shift. $N \times T$ data points are recorded in the table.

Shift	1	2	3	...	N
1	1	1	2	...	2
2	1	1	1	...	1
3	2	1	1	...	1

Table 5.1: Crime data

The second category is the patrol allocation schedule at these shifts. The snapshot of such data is shown in Figure 5.2. We ignore the individual difference between officers and assume that the officers are homogeneous and have the same effect on criminals' behavior. Therefore, only the number of officers at each target and shift affects criminals' behavior and we can summarize the patrol data in the similar manner as crime reports, which is shown in Table 5.2.

Shift	1	2	3	...	N
1	2	1	1	...	1
2	1	1	2	...	2
3	2	1	1	...	1

Table 5.2: Patrol data

Given the available data for crime and patrol officers, our goal is to recommend efficient patrol strategies to prevent opportunistic crimes in problems with a large number of targets. To begin with, we learn criminals' behavior from data and apply the abstract game framework to hierarchically learn the criminal's behavior. Next, we propose a planning algorithm that generates the mixed strategy that optimizes the defender's utility against the learned behavior of criminals.

5.2 Abstract Game

Even though previous approaches [60] deal with opportunistic crimes, they cannot directly be applied to large scale problems. There are two reasons. First, over-fitting is inevitable in the learning process of large scale problems. The number of unknown variables in the learning process is $O(N^2)$ while the number of data points is $O(N \times T)$ [60]. When N increases, the number of variables gets close to the number of data points and causes over-fitting. The second reason is the runtime. The complexity of previous approaches is at least $O(N^{C+1}T)$ where C is the largest value that any variable in the model of [60] can take and it grows quickly with N. In fact, our experiments shows that the algorithm does not converge in one day even with $N = 25$. Therefore, we propose the abstract game framework to deal with opportunistic crimes in large scale urban areas.

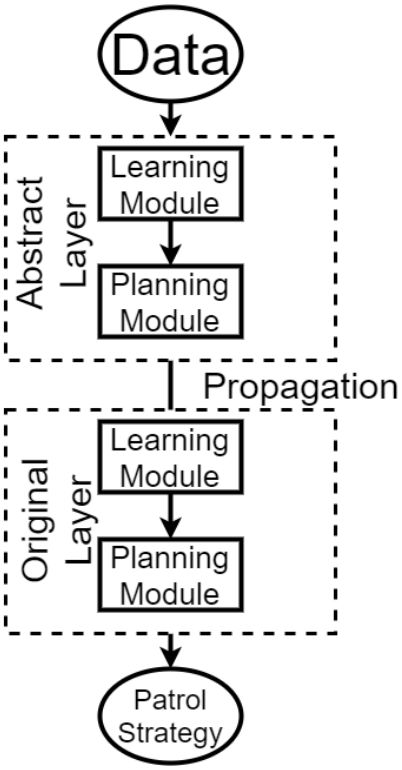


Figure 5.3: Game Abstraction

The idea of abstracting the most essential properties of a complex real problem to form a simple approximate problem has been widely used in the poker game domain [23]. Using such an abstraction the problem can be solved hierarchically and a useful approximation of an optimal strategy for the real problem is provided. In my thesis, we use the concept of abstraction to transform the large scale urban area problem into a smaller abstract problem and solve it hierarchically. Figure 5.3 illustrates the four steps in our game abstraction framework.

First, we need to generate the *abstract layer* from the *original layer* (Section 4.1). Targets that have similar properties are merged together into *aggregated targets*. The set of aggregated targets is called the *abstract layer* while the set of original targets is called the *original layer*. Currently we only consider two layers: the original layer and the abstract layer. If the problem in the abstract layer is still too large to solve, we need to do further abstraction, which we will discuss in Section 4.5. After we obtain the abstract layer, the second step is to learn the criminal's behavior and generate an optimal patrol strategy in the abstract layer (Section 4.2). The third step is to propagate information, such as criminal behavior features, from the abstract layer to the original layer (Section 4.3). Finally, we use the information from the abstract layer and data in the original layer to learn the criminal's behavior and generate an optimal patrol strategy in the original layer (Section 4.4).

5.2.1 Layer Generating Algorithm

We model the layer generation as a Districting Problem [15, 29]. The districting problem is the well known problem of dividing a geographical region into balanced subregions with the notion of balance differing for different applications. For example, police districting problems focus on workload equality [15]. Our layer generation is a districting problem that group targets in the original layer into *aggregated targets* in the abstract layer. However, distinct from the classic Districting problem where the resources are balanced among different *aggregated targets*, in our problem, we try to maximize the similarity of the targets inside the same *aggregated target*. We do so by modeling the similarity

of targets within each aggregated target and use this similarity measure as one of the criteria in the optimization formulation of our problem.

When generating the aggregated targets, there are three principles to follow. First, the aggregated targets should follow the geometric constraints in the districting problem such as contiguity, compactness and environmental constraints. Contiguity means that every target is geographically connected; compactness means that all targets in an aggregated target should be close together; and environmental constraints are the constraints for defender's patrol convenience. For example, if two neighboring targets are divided by a highway, they should not be merged together. Second, the dissimilarity within the aggregated targets should be minimized. We consider two properties of target i, the number of crimes per shift with the defender's presence c_1^i and the number without the defender's presence c_0^i. For target i and target j, we define the Dissimilarity distance function as $Dis_{ij} = |c_1^i - c_1^j| + |c_0^i - c_0^j|$.

Third, the algorithm should consider the scalability constraint for learning algorithm. Let N denote the number of targets in the original layer and n denote the largest scale of problem that the learning and planning algorithms can scale up to. Then there can be no more than n targets inside each *aggregated target* and no more than n *aggregated targets* in the abstract layer. Therefore, $N \leq n^2$ in the original layer. When $N > n^2$, we need multiple layer abstraction that will be introduced later. As we prove next in Lemma 5.2.1, the more the aggregated targets are in the abstract layer, the less information is lost during the abstraction. Hence, we would want to have as many targets as possible in the abstract layer. Thus, we set n aggregated targets in the abstract layer.

Let $I = \{1, \ldots, N\}$ be a set of targets in the original layer. A partition of size K of this set I is a collection of sets $\{I_k\}_{k=1}^K$ such that $I_k \neq \emptyset$ for all $k \in \{1, \ldots, K\}$, $I_k \cap I_l = \emptyset$ for all $k, l \in \{1, \ldots, K\}, k \neq l$ and $\bigcup_{k=1}^K I_k = I$. $\{I_k\}_{k=1}^K$ is the set of the aggregated targets in the abstract layer. Let $\mathcal{P}_K(I)$ denote the set of all partitions of I of size K. Given $I_k \subset I$ we define its inner Dissimilarity as $Dis(I_k) = \sum_{i,j \in I_k} Dis_{ij} = \sum_{i,j \in I_k} |c_1^i - c_1^j| + |c_0^i - c_0^j|$. Also we define its Inertia as $In(I_k) = \min_j \sum_{i \in I_k} d_{ij}$, with d_{ij} denoting the physical distance between the geometric centers of targets i, j. In our districting process we want to find a partition which achieves both low inner Dissimilarity and Inertia over all elements of the partition. Given $\alpha > 0$ as a normalization parameter, we define the information loss function $L_I(K)$ as the lowest cost with a partition of size K, mathematically $L_I(K) = \min_{\{I_k\}_{k=1}^K \in \mathcal{P}_K(I)} \sum_{k=1}^K \alpha In(I_k) + Dis(I_k)$.

Lemma 5.2.1. *The information loss decreases with K, that is $L_I(K+1) \leq L_I(K)$.*

Proof:

First, note that $In(\{i\}) = Dis(\{i\}) = 0$ for all $i \in I$. Let j^* be the value of j that achieves the minimum in $In(I_k) = \min_j \sum_{i \in I_k} d_{ij}$. Let $\{I_j^*\}_{j=1}^K$ the optimal partition of I and $k^* \in \text{argmax}_{k=\{1,\cdots,K\}} \alpha In(I_k^*) + Dis(I_k^*)$. Then $|I_{k^*}^*| > 1$ otherwise $L_N(K) = 0$ and $|I_k^*| = 1$ for all k and there is no clusters. Let $i^* \in I_{k^*}^* - \{j^*\}$. Note that:

$$\begin{aligned} In(I_{k^*}^*) &= \sum_{i \in I_{k^*}^*} d_{ij^*} \geq \sum_{i \in I_{k^*}^* - \{i^*\}} d_{ij^*} \geq In(I_{k^*}^* - \{i^*\}) \\ Dis(I_{k^*}^*) &= \sum_{i,j \in I_{k^*}^*} Dis_{ij} \geq \sum_{i,j \in I_{k^*}^* - \{i^*\}} Dis_{ij} = Dis(I_{k^*}^* - \{i^*\}) \end{aligned}$$

Then,

$$\begin{aligned} L_N(K) = & \sum_{j=1}^{K} \alpha In(I^*_{k^*}) + Dis(I^*_{k^*}) \\ \geq & \sum_{j=1}^{K} \alpha In(I^*_{k^*}) + Dis(I^*_{k^*}) \\ & + \underbrace{\alpha In(\{i^*\}) + Dis(\{i^*\})}_{=0} \\ \geq & L_N(K+1) \end{aligned}$$

Then, the partition $\{I_1, \ldots, I_{k^*-1}, I_{k^*} - \{i^*\}, \{i^*\}, I_{k^*+1}, \ldots, I_K\} \in \mathcal{P}_I(K+1)$ is feasible for the problem of $K+1$ clusters and has lower loss function value, then the optimal clustering in $\mathcal{P}_I(K+1)$ also has the lower objective function.

Based on these three principles, we propose a mixed integer linear program (MILP) to solve the districting problem. We apply an extension of the capacitated K-median problem with $K = n$. While the capacitated K-median problem [50] satisfies the scalability constraint by setting a maximum capacity for each aggregated target, it cannot handle the geometric constraints such as contiguity. A counterexample is shown below. In this work, we handle the geometric constraints by considering the inertia of each aggregated target as part of the information loss function.

Counterexample of capacitated K-median problem

For a problem with 4 targets located on a 1-dimension string. Target 1 has attractiveness 0.9 and geometric information [0,0.001]; Target 2 has attractiveness 0.1 and geometric information [0.001,0.002]; Target 3 has attractiveness 0.9 and geometric information [0.002,0.003]; Target 4 has attractiveness 0.9 and geometric information [0.003,0.004]. Using capacitated K-median problem, target 1 and 3 should be clustered together while

2 and 4 should be clustered together. However, such segmentation violate the contiguity constrained in the Districting problem so the approach is inapplicable in the problem.

$$
\begin{aligned}
\min_{x,y,z} \quad & \alpha \textstyle\sum_{i,j} d_{ij} y_{ij} + \sum_{ik} z_{ik} & \\
s.t. \quad & \textstyle\sum_j y_{ij} = 1 & \forall i \in I \\
& y_{ij} \leq x_j & \forall i,j \in I \\
& \textstyle\sum_j x_j = n & \\
& \textstyle\sum_j y_{ij} \leq n & \forall j \in I \\
& z_{ik} \geq Dis_{ik}(y_{ij} + y_{kj} - 1) & \forall i,k,j \in I \\
& z_{ik} \geq 0 & \forall i,k \in I \\
& y_{ij} + y_{kj} \leq 1 & \forall j \in I \\
& y_{ij}, x_j \in \{0,1\} & \forall i,j \in I
\end{aligned}
\tag{5.1}
$$

x_j is a binary variable. It is 1 if the target j is the center of an aggregated target and 0 otherwise. The variable y_{ij} takes the value 1 when the target i is allocated to the aggregated target centered in j and 0 otherwise. The variable z_{ik} is a continuous non-negative variable that takes the value Dis_{ik} when target i and target k are allocated to the same aggregated target, otherwise z_{ik} is 0. The objective function is the weighted sum of inertia and dissimilarity. α represents the trade-off between geometric shape and the similarity within each aggregated target.

The first set of constraints ensures that every target is allocated to an aggregated target. The second set of constraints ensures that the center of an aggregated target belongs to this aggregated target. The third expression states that there are n aggregated

targets. The fourth set of inequalities ensures the size of every aggregated target to be no greater than n. The fifth and sixth constraint ensures that z_{ik} will take the value Dis_{ik} when target i and target k are allocated to the same aggregated target, otherwise z_{ik} will be 0. The seventh constraint is an example of environmental constraints that target i and target k cannot be in the same aggregated target.

Directly solving this MILP is NP-hard [30]. Therefore we use the heuristic constraint generation algorithm (Algorithm 6) to approximately solve the problem.

Input: I: Set of targets; K: number of aggregated targets
Output: $\{y^*\}$: objective_ function.
1 Center $\leftarrow$ Location_Problem(I, K); Cuts=$\emptyset$ **for** $i = 1, \cdots, MAX\_IT$ **do**
2 $y^*, z^* \leftarrow$ Allocation_Phase(Center, Cuts)
$i^*, j^*, k^* = \text{argmin}_{i,j,k}\, z^*_{ik} - Dis_{ik}(y_{ij} + y_{kj} - 1)$ **if**
$(z^*_{i^*k^*} - Dis_{i^*k^*}(y_{i^*j^*} + y_{k^*j^*} - 1)) \geq 0$ **then**
3 break
4 **end**
5 **else**
6 Cuts $\leftarrow$ Cuts $\cup \{z_{i^*k^*} \geq Dis_{i^*k^*}(y_{i^*j^*} + y_{k^*j^*} - 1)\}$
7 **end**
8 **end**

Algorithm 6: Constraint Generation Algorithm

The algorithm has two phases: first, the location problem is solved as a K-median problem. In the second phase, we use the constraint generation technique [8] to solve the optimization problem. The iterative constraint generation algorithm is shown as the for loop (line 2-9). To start with, all the constraints $z_{ik} \geq Dis_{ik}(y_{ij} + y_{kj} - 1)$ for i, j, k are removed completely (denoted by the empty set $Cuts$ in line 1), and then in each iteration of the for loop the MILP is solved (line 3) and then we check whether any of the left out constraints are violated (line 4, 5). If yes, then the most violated constraint is added to

Cuts or else the loop stops. The maximum number of iterations is limited by MAX_IT. Constraint generation guarantees an optimal solution given large enough MAX_IT.

5.2.2 Abstract Layer

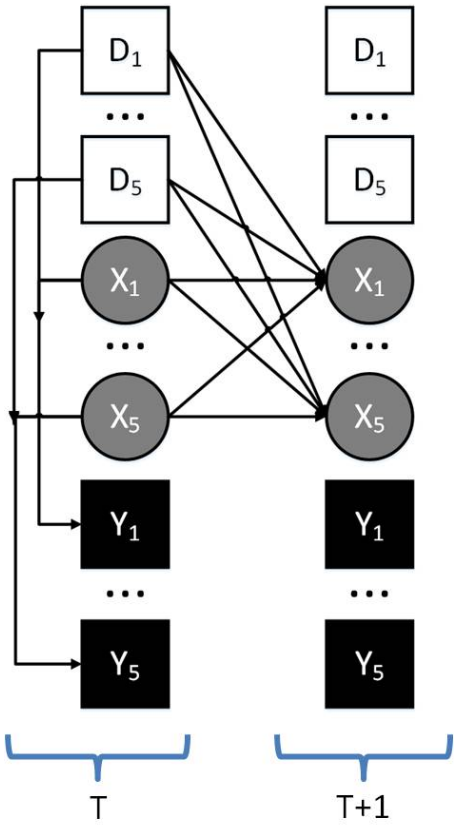


Figure 5.4: DBN framework

Learning Algorithms: As noted earlier, having generated the abstract layer, the next step is learn the adversary model at the abstract layer. As stated before, the Dynamic Bayes Network (DBN) learning algorithm presented in [60] could not be used in the original layer due to scaling difficulties; however, with a sufficiently small number of targets in the abstract layer, we can now use it. To illustrate its operation, we reproduce the operation with N targets as shown in Figure 5.4. Three types of variables are considered in the DBN: squares in the top represent the number of defenders at aggregated target i during shift t, $D_{i,t}$, squares in the bottom represent the number of crimes at aggregated target i during shift t, $Y_{i,t}$, while circles represents the number of criminals at aggregated target i during shift t, $X_{i,t}$. As shown in Figure 5.4, there are two transitions in the DBN: the criminal's transition from shift t to $t+1$, which is modeled as the transition probability and the crime transition at shift t, which is modeled as the crime output probability. Mathematically, a transition probability is defined

as $P(X_{i,t+1}|D_{1,t},...,D_{N,t},X_{1,t},...,X_{N,t})$ and the crime output probability is defined as $P(Y_{i,t}|D_{1,t},...,D_{N,t},X_{1,t},...,X_{N,t})$. This model uses two matrices to represent the transition probabilities, the movement matrix A which consists of all the criminal's transition probability $P(X_{i,t+1}|D_{1,t},...,D_{N,t},X_{1,t},...,X_{N,t})$ and the crime matrix B which consists of all the crime output probability $P(Y_{i,t}|D_{1,t},...,D_{N,t},X_{1,t},...,X_{N,t})$. A and B contains $C^N \times C^N \times C^N$ unknown parameters.

Given available data about $D_{i,t}$ (patrol schedule), $Y_{i,t}$ (crime report), this model applies the Expectation Maximization algorithm to learn A and B while estimating $X_{i,t}$. The detail of this learning model is present in [60]. The novelty in this chapter is propagating adversary behavior parameters (A and B) from the abstract layer to the original layer, which we discuss in Section 4.3; but we do that, we discuss planning in the abstract layer.

Planning Algorithms: In this chapter, we focus on planning with mixed strategies for the defender rather than the pure strategy plans from previous work [60]. This change in focus is based on two key reasons. First, this change essentially broadens the scope of the defender's strategies; if pure strategies are superior our new algorithm will settle on those (but it tends to result in mixed strategies). Second, previous work [60] on planning with pure strategies depended on repeatedly cycling through the following steps: planning multiple shifts of police allocation for a finite horizon, followed by updating of the model with data. This approach critically depended on the model getting updated periodically in deployment. Such periodic updating was not always easy to ensure. Thus, within any one cycle, the algorithm in [60] led to a single pure strategy (single police allocation) being repeated over the finite horizon in real-world tests as it tried to act based on the

model learned from past data; such repetition was due to a lack of updating of the criminal model with data, and in the real-world, the criminals would be able to exploit such repetition. Instead, here we plan for a mixed strategy. We assume that the model updates may not occur frequently and as a result we plan for a steady state.

We model the planning procedure as an optimization problem where our objective is to maximize the defender's utility per shift. After the defenders' (mixed) strategy is deployed for a long time, criminals receive perfect information of the strategy and their (probabilistic) reaction will not change over time. As a result, the criminals' distribution becomes stationary and this is called criminals' stationary state. In our case, ergodicity guarantees unique stationary state.

Ergodic to unique stationary distribution

For criminal's distribution, it is a Markov Chain. This Markov chain is ergodic since the criminal's distribution can transfer from any state to any other states. Therefore, this Markov chain is ergodic. By the Fundamental Theorem of Markov Chain, a ergodic Markov chain P has a unique probability vector $\mathbf{c}$ such that $P \cdot \mathbf{c} = \mathbf{c}$. $\mathbf{c}$ is the stationary distribution of criminal. Therefore, ergodic leads to unique stationary distribution.

Crime and stationary constraints

As introduced in [60], $P(Y_{i,t}|d_1, ..., d_N, x_1, ..., x_N)$ is defined as the crime probability, which can be represented as $P(Y_{i,t}|d_1, ..., d_N, x_1, ..., x_N) = B(Y_i, X_i, D_i, i)$. At the same time, $P(X_{i,t+1}|d_1, ..., d_N, x_1, ..., x_N)$ is defined as the criminal transition probability, which can be represented as $A(x_{i,t+1}, d_{1,t}, ..., d_{N,t}, x_{1,t}, ..., x_{N,t})$. Therefore, the crime and stationary constraints can be represented as below:

$$
\begin{aligned}
& y_i = \textstyle\sum_{Y_{i,t}} Y_{i,t} \cdot \\
& B(Y_i, X_i, D_i, i),\ i \in I, \\
& x_i = \textstyle\sum_{X_{i,t+1}} X_{i,t+1} \cdot \\
& A(x_{i,t+1}, d_{1,t}, ..., d_{N,t}, x_{1,t}, ..., x_{N,t}),\ i \in I.
\end{aligned}
\tag{5.2}
$$

Reconstruct transition matrices from human behavior parameters

Given Equation 3 in Section 4.3, where we extract behavior parameters from A and B. We can inverse this process and get A and B from λ, μ and Att as below:

$$
\begin{aligned}
A(X_{j,t+1}, D_{i,t}, X_{i,t}) &= \begin{cases} \frac{e^{Att_j^{new}}}{\sum_{n \in N} e^{Att_n^{new}}} \cdot e^{\lambda_i X_{i,t} + \mu_i D_{i,t}}, & \text{if } i \neq j \\ \frac{e^{Att_j^{new}}}{\sum_{n \in N} e^{Att_n^{new}}} \cdot e^{\lambda_i X_{i,t} - \mu_i D_{i,t}}, & \text{otherwise} \end{cases} \\
B(Y_{i,t}, D_{i,t}, X_{i,t}) &= B(Y_{k,t}, D_{k,t}, X_{k,t}) \cdot \frac{Att_i^{new}}{\sum_{i \in k} Att_i^{new}}
\end{aligned}
\tag{5.3}
$$

where k is the aggregated target that target i belongs to.

Our planning algorithm assumes criminals' stationary state when maximizing the defender's utility. We define defender's utility as the negation of the number of crimes. Therefore, the objective is to minimize the number of crimes that happen per shift in the stationary state. Let's define $I = \{i\}$ as the set of aggregated targets, D as the total number of defenders that are available for allocation; $d_I = \{d_i\}$ as the set of defender's allocation at target set I, $x_I = \{x_i\}$ as the set of criminal's stationary distribution at target set I with respect to defender's strategy d_I and $y_I = \{y_i\}$ as the set of expected number of crimes at target I. Note that C is the largest value that the variables D_i, X_i and Y_i can take. The optimization problem can be formed as follows:

$$
\begin{aligned}
\underset{d_I}{\text{minimize}} \quad & \textstyle\sum_{i \in I} y_i \\
\text{subject to} \quad & 0 \leq x_i \leq C,\ i \in I, \\
& 0 \leq d_i \leq C,\ i \in I, \\
& \textstyle\sum_{i \in I} d_i \leq D, \\
& y_i = \textstyle\sum_{Y_{i,t}} Y_{i,t}. \\
& P(Y_{i,t} | d_1, ..., d_N, x_1, ..., x_N),\ i \in I, \\
& x_i = \textstyle\sum_{X_{i,t+1}} X_{i,t+1}. \\
& P(X_{i,t+1} | d_1, ..., d_N, x_1, ..., x_N),\ i \in I.
\end{aligned}
\tag{5.4}
$$

In this optimization problem, we are trying to minimize the total number of crimes occurring in one shift while satisfying five sets of constraints. The first two constraints ensure the defender and criminal's distribution are non-negative and no more than an upper bound C. The third constraint represents the constraint that the number of deployed defender resources cannot be more than the available defender resources. The fourth constraint is the crime constraint. It sets y_i to be the expected number of crime at target i. The last constraint is the stationary constraint, which means that the criminals' distribution is not changing from shift to shift with respect to the patrol strategy d_I. The transitions are calculated by movement matrix A and crime matrix B. When merging targets in layer m to generate targets in layer $m+1$, we need to figure out the total number of targets in layer $m+1$. The direct approach to solve this problem is to enumerate all the possible combinations of number of targets in all M layers. This

approach is computationally inapplicable since we have to call the MILP in Section 4.1 for $o(N^M \cdot M)$ times.

For example, if $N = 50$ and $n = 5$. Then, there should be $M = log_5 50 + 1 = 3$ layers. For layer 1, there will be 50 targets. For layer 2, the number of targets can be from 10 to 25. For layer three, the number of targets can be 2 to 5. The direct learning algorithm tries every possible combinations of these three layers and runs the MILP for each combination to generate the optimal segmentation with respect to the combination. For example, we run MILP for $50 \rightarrow 25 \rightarrow 5$, $50 \rightarrow 24 \rightarrow 5 ... 50 \rightarrow 10 \rightarrow 2$.

In order to solve the problem, we first propose a Dynamic Programming layer generating algorithm so that the minimum information lost is achieved by abstraction. We introduce this algorithm by mathematical induction.

In layer 1, there are N targets. In layer 2, the possible number of targets N_2 has a range of $\frac{N}{n} <= N_2 <= n^{M-1}$. The lower bound for N_2 ensures the *aggregated targets* in layer 2 does not includes more than n targets in layer 1; the upper bound is the necessary condition that in layer $m > 2$, no aggregated targets includes more than n targets. For all the possible N_2, we run the MILP in Section 4.1. Then we get the target clusters and information loss for all N_2, denoted as $Cl(N_2)$ and $Los(N_2)$. The number of calls is $o(N)$.

Assume for mth layer, the number of targets ranges $[N_m^l, N_m^u]$, then for $(m+1)$th layer, the number of targets N_{m+1} ranges from $\frac{N_m^l}{n}$ to $min(\frac{N_m^u}{n}, n^{M+1-m})$. For each possible N_{m+1} in layer $m+1$, we enumerate N_m in layer m. If $N_m < n \cdot N_{m+1}$, we run the layer generation algorithm in Section 4.1 from N_m to N_{m+1} and get the clusters and information losses. We denote information loss from N_m to N_{m+1} as $Los(N_m, N_{m+1})$ while clusters as $Cl(N_m, N_{m+1})$. We choose the minimum loss among all $Los(N_m, N_{m+1})$

as $Los(N_{m+1})$ and the corresponding cluster as $Cl(N_{m+1})$. The number of MILP calls is $o(N^2)$.

By repeating the above process $M-2$ times, we can get the optimal clusters for M layers. The mathematical formulation is shown in Algorithm 7. In this algorithm, the optimal clusters are achieved and we only need to call the MILP $O(N^2 \cdot M)$ times, which is significantly smaller than direct approach.

```
1 M = ⌊log_n N⌋ + 1, N_1^min = N, N_1^max = N for m = 2, ··· , M do
2   N_m^l = N_{m-1}^l / n, N_m^u = min(N_{m-1}^u / n, n^{M-m}) for i = N_m^l, ..., N_m^u do
3     opt_objective(i) = +∞, Targets(i) = null, optimal_path_to(i)=null for
        j = N_{m-1}^l, ..., N_{m-1}^u do
4       Cluster(i), obj(i)= Cluster_Algorithm(Cluster(j), i) if opt_objective(i) >
          opt_objective(j)+obj(i) then
5         opt_objective(i)= opt_objective(j)+obj(i) optimal_path_to(i) = j
            Targets(i) = I(i)
6       end
7     end
8   end
9 end
```

Algorithm 7: Dynamic Programming based Multi-Layer Generating Algorithm

Another layer generating algorithm is the greedy algorithm. By Lemma 1, we know that the more targets in that layer, the less information is lost at this layer. Therefore, the greedy algorithm set the number of targets to be maximum. Therefore, for the mth layer, the number of targets is n^{M+1-m}. The number of calls is only $o(M)$. However, the clusters are only local optima.

5.2.3 Propagation of learned criminal model

In the previous section, we generate the patrol allocation for the aggregated targets in the abstract layer. In order to provide patrolling instructions for the original layer, we

propagate the learned criminal model in the abstract layer to the original layer. We need to address two cases: when there is no detailed patrol data and when there is. In particular, we have found that some police departments record the location of police patrols in detail at the level of targets in the original layer, but many others specifically only keep approximate information and do not record details (even if they record all crime locations in detail); thus leading to the two cases. We start by describing the case with sufficient patrol data in the original layer.

Direct learning (sufficient data): When there is detailed patrol data in the original layer and nothing is approximated away, we know the numbers of police at each target in the original layer at each shift. Then, we can directly learn A and B in this DBN. The learning algorithm is same as that applied in the abstract layer. The data used in the algorithm is the crime report and patrol schedule inside each *aggregated target*. While we directly learn A and B, computation of the patrol strategy at the abstract layer affects the patrol strategy in the detailed layer as discussed in Section 5.2.4.

Parameter Propagation (limited data): If the patrol data in the original layer is limited, the DBN model that we learned in the original layer will be inaccurate if we still apply the same learning algorithm as the abstract layer to learn matrices A and B. One remedy is to provide addition criminal information to the original layer from the abstract layer to help the process of learning criminal model in the original layer. However, in the abstract layer, movement matrix A and crime matrix B represent the criminal's behavior in aggregated targets. It cannot directly describe the criminal's behavior in the targets in the original layer. Therefore, we propose a human behavior based model of extracting behavior parameters from A and B in the abstract layer. Then we set these behavior

parameters of an aggregated target as the behavior parameters for the targets contained within this aggregated target in the original layer.

Parameter extraction: We introduce the process of using a human behavior model to extract the behavior parameters from A and B. The basic assumption of a human behavior model is that the criminal follows certain patterns when moving from shift to shift. Specifically, the criminals follow the movement by the well established Quantal Response (QR). In the learning algorithm [60], one simplification made was breaking down the criminals' transition probabilities into marginal probability $P(X_{j,t+1}|D_{i,t}, X_{i,t})$ which represents the movement of a criminal from target i to target j. Based on the Quantal Response model, we approximate this movement using the following equation: $P(X_{j,t+1} = 1) = \frac{e^{Att_j}}{\sum_{n \in N} e^{Att_n}}$ where Att_n is the attractiveness property of target n. In the DBN, the movement depends not only on the attractiveness, but also on the allocation of defenders and criminals at previous shift. Therefore, we formulate $\hat{P}(X_{j,t+1} = 1|D_{i,t}, X_{i,t})$ as $(\lambda_i, \mu_i \geq 0)$:

$$\begin{cases} \frac{e^{Att_j}}{\sum_{n \in N} e^{Att_n}} \cdot e^{\lambda_i X_{i,t} + \mu_i D_{i,t}}, & \text{if } i \neq j \\ \frac{e^{Att_j}}{\sum_{n \in N} e^{Att_n}} \cdot e^{\lambda_i X_{i,t} - \mu_i D_{i,t}}, & \text{otherwise} \end{cases} \tag{5.5}$$

The reason for the above effect of defender is that the defender at target i disperses criminals to other targets. However, λ, μ and Att are not known and we need to learn them from data. Our approach to compute λ, μ and Att is to find their values that minimize the L_1 distance between $\hat{P}(X_{j,t+1} = 1|D_{i,t}, X_{i,t})$ and the learned marginal probability $P(X_{j,t+1} = 1|D_{i,t}, X_{i,t})$. We can formulate this problem as the following optimization:

$$\min_{Att,\lambda,\mu} \sum_{i,j,D_{i,t},X_{i,t}} \quad ||P(X_{j,t+1} = 1|D_{i,t}, X_{i,t}) - \hat{P}(X_{j,t+1} = 1|D_{i,t}, X_{i,t})||$$

$$\text{subject to} \quad \mu_i \geq 0, \lambda_i \geq 0, i = 1, ..., N$$

The constraints represent the positive effect of number of criminals on the transition probability and more defenders lead to faster dispersion of criminals. λ, μ and Att are the behavior parameters that we propagate to original layer.

Since λ and μ represent the influence of the number of criminals and number of defenders on the criminals' movement in the aggregated target, it is reasonable to assume that the criminals' movement in the targets that belong to the aggregated target inherit these parameters. In other words, this means that the influence of the number of criminals and defenders is the same within the aggregated target. At the same time, Att measures the availability of the crime opportunities. Therefore, within one aggregated target, the attractiveness is distributed among the targets proportional to the total number of crimes in each target. For example, if the attractiveness of an aggregated target I (made up of I_1 and I_2) is 0.6, the total number of crimes at target I_1 is 80 while that at target I_2 is 40, then the attractiveness of A_1 is 0.4 while that of A_2 is 0.2. λ, μ and Att for each target are the behavior parameters that will be used in crime and stationary constraints in the planning algorithm.

5.2.4 Computing Strategy in the Original Layer

In the previous section, we generated the adversary behavior parameters in the original layer. In order to provide patrolling instructions for the original layer, we utilize the strategy in the abstract layer to assign resources in the original layer. Then, combined with the propagated adversary behavior parameters we generate the strategy at the original layer.

Resource Allocation: In the abstract layer, the optimal strategy recommends the number of resources allocated to each *aggregated target*. We use this recommendation as a constraint on the number of resources in planning within the *aggregated targets* at the original layer. For example, the abstract layer may provide 0.8 as the allocation to an aggregated target say X; then we plan patrols in X in the original layer using 0.8 as the total number of resources.

Next, in the original layer, we treat each *aggregated target* in the abstract layer as an independent DBN as shown in Figure 5.4. The same algorithm for generating a mixed strategy in the abstract layer can be applied in each of the independent DBNs. The optimization problem is the same as Equation 5.4. D is the total number of resources allocated to these aggregated targets (e.g., 0.8 to target X).

In addition, the formulations of crime and stationary constraints required in the computation of the mixed strategy are different for the scenario with sufficient and limited data. For the scenario with sufficient data these constraints are formulated using the parameters A and B of the DBN that is learned in this original layer. For the scenario with limited data the propagated values of λ, μ and Att are used to estimate the the A

and B parameters for the DBN representation of the adversary behavior in the original layer. The estimation is the inversion of parameter extraction, and it happens in the original layer. For example, we use Equation 5.5 to estimate the parameters using λ, μ and Att. The details are presented in the appendix. Then, these reconstructed A and B are used to formulate the crime and stationary constraints.

5.2.5 Extended Abstract Game

When $n^2 < N$, we can use two layers of abstraction to solve the problem. However, when the real problem has $N > n^2$ targets, even two layered abstraction does not suffice since there must be a layer in the game with more than n targets. Therefore, we propose the multiple layer framework to handle problems with an arbitrarily large number of targets. This framework is an extension of the two layer abstract game. We apply an iterative four step process. As a first step, we need to decide the number of layers as well as the districting of targets for each of the layers. Considering the scalability constraints (recall that there cannot be more than n targets within each aggregated target), the number of layers is $M = \lfloor \log_n N \rfloor + 1$. We denote the original layer as Layer 1 and the layer directly generated from Layer m as layer $m + 1$. In this notation, the topmost abstract layer is Layer M. The second step is learning criminals behavior in the top layer. The third step is to generate a patrol strategy at this layer. The fourth step is to propagate parameters to the next layer. We keep executing steps two to four for each layer until we reach the original layer. At each layer, we decide whether to do parameter propagation based on the availability of the patrol data. If we have sufficient patrol data at layer m, we do

direct learning at layer m. Otherwise, we do parameter propagation from layer $m+1$ to layer m.

We propose three different layer generation algorithms. The first algorithm is the direct algorithm. For example, if $N = 50$ and $n = 5$. Then, there should be $M = 3$ layers. For layer 1, there will be 50 targets. For layer 2, the number of targets could be any integer between 10 to 25. For layer 3, the number of targets can be 2 to 5. The direct learning tries all the combinations of three layers and runs the MILP for each combination to generate the optimal segmentation. It calls the MILP in Section 4.1 for $O(N^M \cdot M)$ times; the second algorithm is a dynamic programming approach that ensures the solution is globally optimal. The MILP is called $O(N^2 \cdot M)$ times; the third algorithm is the greedy algorithm that sets the number of targets to be maximum, which for the mth layer is n^{M+1-m}. The number of calls is M while the solution is not necessarily optimal. Details are in the appendix.

5.3 Real World Validation

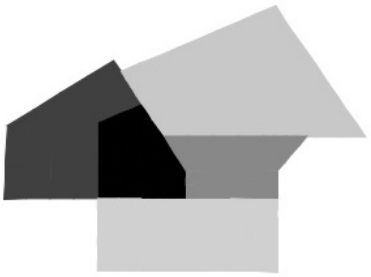


Figure 5.5: Campus map 1

Figure 5.6: Campus map 2

We use two sets of real world data to validate the game abstraction framework. In the first case we use the data from the University of Southern California (USC) campus

that is provided by [60]. We thank the authors for providing three years (2012-2014) of crime report and patrol schedule from the USC campus. The number of total crime events is on the order of 10^2. [60] reports that the campus patrol area (USC campus and its surroundings) is divided into five patrol areas, which are shown in Fig 5.5. In order to make the patrols more efficient, the police officers wish to further divide the whole campus into 25 patrol areas and get patrol recommendations on these 25 patrol areas. There are two tasks for us, (a) starting from city blocks (there are 298 city blocks and they form the basis of the USC map), create 25 separate "targets", as in our layer generation problem; (b) generate an optimal patrol strategy for these 25 targets. The creation of 25 targets is also a districting problem and the technique in Section 4.1 can be directly applied. The 25 targets generated by the districting algorithm is shown in Figure 6.1. We treat these 25 targets as the original layer. n is set to be 5 as the runtime of learning and planning algorithm with $n = 5$ is reasonably small. So then we use two layer game abstraction to solve this problem with 25 targets. The abstract layer is the five patrol areas in Fig 5.5. This is because of the center area (the darkest area) is the campus itself and is separated from its environment by fences and gates. These environmental constraints cause our layer generation to automatically create the area into 5 targets as shown in Figure 5.5. Additionally, police only record their presence in the five areas, and thus, we do not have detailed police presence data; as a result, we use our behavior learning to propagate parameters from the abstract layer to the original layer.

In the second case, we use data about crime and detailed police patrol locations in Nashville, TN, USA. The data covers a total area of 526 sq. miles. Only burglaries (burglary/breaking and entering) have been considered for the analysis. Burglary is the

chosen crime type as it is a major portion of all property crimes and is well distributed throughout the county. Data for 10 months in 2009 is used. The number of total crime events is on the order of 10^3. Observations that lacked coordinates were geocoded from their addresses. Police presence is calculated from GPS dispatches made by police patrol vehicles. Each dispatch consists of a unique vehicle identifier, a timestamp and the exact location of the vehicle at that point in time. We divide the whole city into $N = 900$ targets as shown in Figure 5.7. Since n is 5, the number of layers we need is $M = \lfloor \log_5 900 \rfloor + 1 = 5$. We use the multiple layer abstraction framework to solve this problem.

5.4 Experimental Results

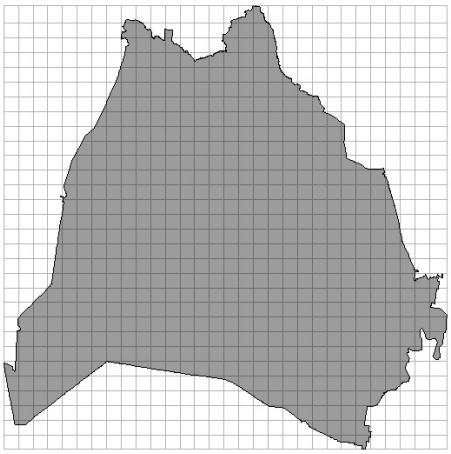


Figure 5.7: City map

Experiment setup. We use MATLAB to solve our optimization problems. There are two threads of experiments, one on the USC campus problem and the other on Nashville, TN problem. To avoid leaking confidential information of police departments, all crime numbers shown in the results are normalized. The experiments were run on a machine with 2.4 GHz and 16 GB RAM.

Game Abstraction Framework: Our first experiment is on comparing the performance of our game abstraction framework with the DBN framework proposed in [60] for large scale problems. Since the DBN framework cannot even scale to problems with 25 targets, in

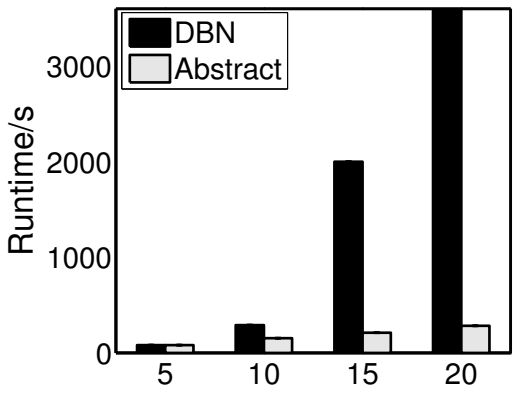


Figure 5.8: Runtime

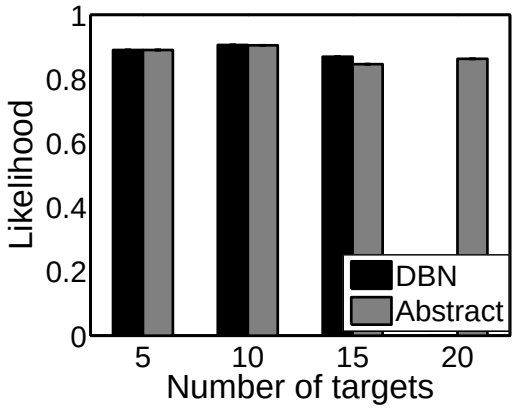


Figure 5.9: Likelihood

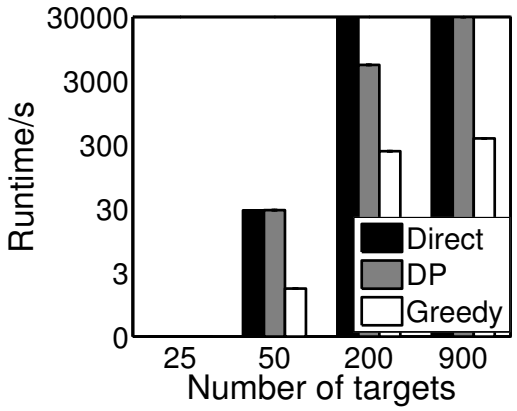


Figure 5.10: Runtime

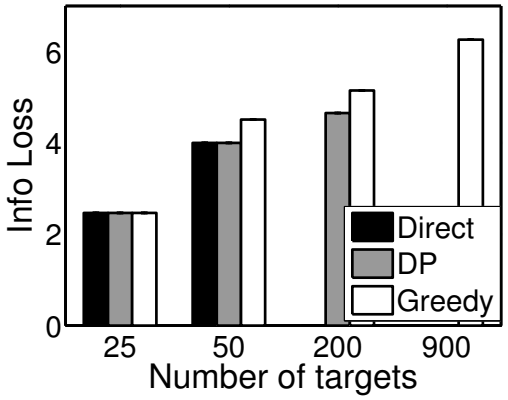


Figure 5.11: Information Loss

this experiment we run on problems with subsets containing N targets ($5 \leq N < 25$) out of these 25 targets in the USC campus. As shown in Figure 5.8, we compare the runtime of these two frameworks. The x-axis in Fig. 5.8 is the number of targets N in the problem. For each N, we try ten different subsets and the average runtime is reported. The y-axis indicates the runtime in seconds. The cut-off time is 3600s. As can be seen in Figure 5.8, the runtime of the DBN framework grows exponentially with the scale of the problem and cannot finish in an hour when $N = 20$. At the same time, the runtime of the game abstraction framework grows linearly with the scale of the problem. It takes less than 5 minutes to solve the problems with $N = 20$. This indicates that the DBN

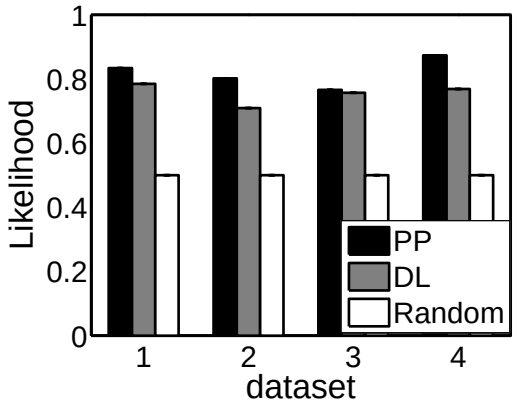


Figure 5.12: Likelihood (USC)

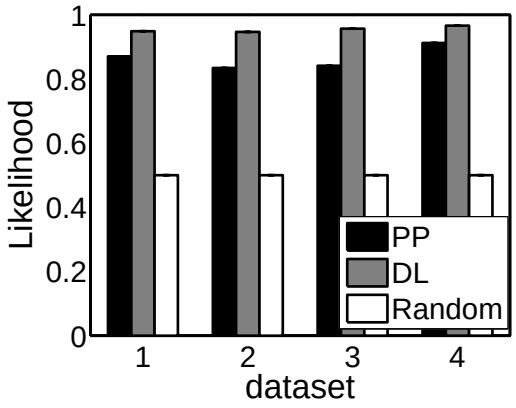


Figure 5.13: Likelihood (city)

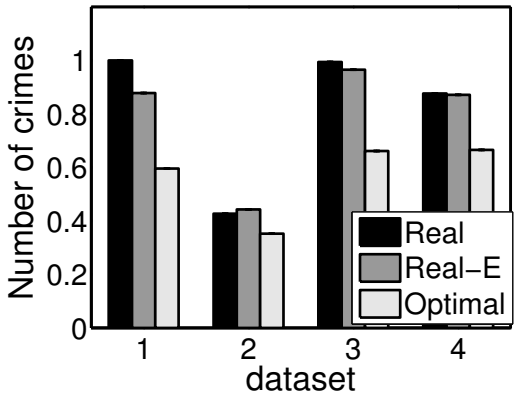


Figure 5.14: Plan (USC)

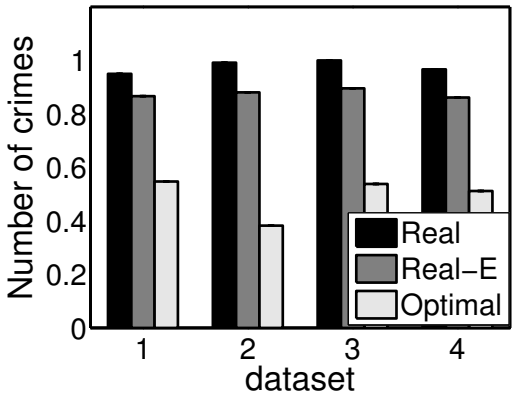


Figure 5.15: Plan (city)

framework fails to scale up to large scale problems while the game abstraction framework can handle many more targets.

In Figure 5.9 we compare the *likelihood* of these two different frameworks. We divide the 36 months' data sets into two parts, the first 35 months' data is used for learning while we predict the crime distribution for the last month and compare it with the real crime data in that month. For every target and every shift, we measure the *likelihood* as the predicted probability of the number of crimes reported in the data for that target and shift. For example, for target i and shift t, our prediction is that there is 30% probability that no crime occurs and 70% that one crime occurs while in the data there

is one crime at target i in shift t. Then, the likelihood for target i for shift t is 0.7. Since in most areas there is no crime, we first selected targets with crime. The we select a random set of targets without crime that has the equal number as those with crime. The reported likelihood is the average likelihood over all selected targets and all shifts over all ten different subsets. The higher the likelihood, the better our prediction. As can be seen in Figure 5.9, the game abstraction framework achieves similar likelihood compared to the DBN algorithms given any number of targets in the problem. This indicates that even through information may be lost during the abstraction, the game abstraction framework captures important features of the criminal and performs as well as the exact DBN framework while running 100 of times faster.

Layer Generation Algorithm: Next, we use the data from the city to evaluate the performance of our layer generation algorithms. Again, we run the layer generation algorithms on problems with subsets containing N targets ($N \leq 900$) out of the 900 targets in the city map. For each N, we try ten different subsets and report the average value except when $N = 900$ for which only one subset is possible. Figure 5.10 compares the runtime of different layer generation algorithms in log format. Three different algorithms are compared, the direct algorithm (Direct) that traverses all possible layer combinations; the dynamic programming algorithm (DP) and the greedy algorithm (Greedy). The x-axis in Fig. 5.10 is the number of targets N. For $N = 25$, two layers are needed; for $N = 50$, three layers are needed; for $N = 200$, four layers are needed and for $N = 900$, five layers are needed. The y-axis is the runtime of different algorithms in seconds. The cut-off time is set at 36000s. When $N = 25$, the runtime of these three algorithms are the same because the layer generation is unique. The number of targets in layer 2 is 5. When

$N = 50$, the runtime of the direct algorithm is the same as that of the DP algorithm while the runtime of the greedy algorithm is significantly lower. When $N = 200$, the direct algorithm cannot finish in 10 hours; the DP algorithm takes around five hours while greedy algorithm finishes in less than 10 minutes. When $N = 900$, both direct learning and DP are cut off while the runtime for greedy is less than 15 minutes. This validates our theoretical result that the runtime of direct algorithm grows exponentially with the scale of the problem, that of DP grows polynomially and that of greedy algorithm grows linearly with the number of layers. Since both direct and DP algorithm cannot scale up to the problem with $N = 900$, we use the greedy algorithm as the layer generation algorithm in the city problem.

In Figure 5.11, we compare the information loss of different layer generation algorithms. The information loss is defined as the objective in Equation 5.1. As can be seen in Fig. 5.11, the information loss of DP is the same as that of direct learning in any situations. This is because DP ensures a globally optimal solution. At the same time, the information loss of the greedy algorithm is higher than that of the DP algorithm but no more than 15% higher. This indicates that while greedy algorithm cannot ensure global optimal information loss, it can reach a good approximation in reasonable runtime.

Learning: Third, we evaluate the performance of our learning algorithm. Game abstraction is used for both problems and we evaluate the predictions in the original layer.

The result shown in Figure 5.12 and Figure 5.13 compares the likelihood of different algorithms in USC campus and the city problem respectively. Three different algorithms are compared: (1) the Random approach, in which the probabilities of each situation

are the same (Random), (2) game abstraction with direct learning for both the abstract and original layer (DL) and (3) game abstraction with parameter propagation in the original layer (PP). We divide the whole data sets into four equal parts. For each part, the first 90% of data is used for training while we test on the last 10% of data. The x-axis in Fig. 5.12 and 5.13 is the index of the part of data that we evaluate on. y-axis indicates the likelihood on the test set. As can be seen in both figures, the likelihood of both game abstraction based approaches are higher than that of the baseline random algorithm in all the test sets. This indicates that game abstraction models help improve the prediction in large scale problems. In addition, parameter propagation at the original layer outperforms direct learning at this layer in the USC problem in Figure 5.12. Direct learning outperforms parameter propagation in Nashville problem in Figure 5.13. This is because the patrol data at the original layer in USC is limited. That is, only the aggregate number of police resources over several targets is available while the resources at each target remain unknown. Parameter propagation is better at handling limited patrol data. However, the patrol data is adequate in the city problem and direct learning is a better fit in such situations. Therefore, in the later experiments, we use parameter propagation as the learning algorithm in the USC and direct learning as the learning algorithm in Nashville.

The result shown in Figure 5.16 compares the likelihood of different layer generation algorithms in the Nashville problem. Two different algorithms are compared: (1) the Optimal approach by our layer generation algorithm (Optimal); (2) the Random approach, in which the neighboring targets are randomly merged to form the new layers (Random). We divide the whole data sets into four equal parts. For each part, the first 90% of data

is used for training while we test on the last 10% of data. The x-axis in Fig. 5.16 is the index of the part of data that we evaluate on. y-axis indicates the likelihood on the test set. As can be seen in Figure 5.16, the likelihood of optimal layer generation is significantly higher than that of the random layer generation. This indicates that optimal layer generation helps improve the crime prediction in large scale abstraction models.

Planning: Next, we evaluate the performance of our planning algorithm in both the problems. Figure 5.14 and 5.15 compare strategies generated using the game abstraction framework with the actual deployed allocation strategy generated by the domain experts. Three different scenarios are compared: the real number of crimes, shown as Real; the expected number of crimes with manually generated strategies and learned adversary model with game abstraction, shown as Real-E and the expected number of crimes with the optimal strategy computed using game abstraction, shown as Optimal. As shown in Figure 5.14 and 5.15, the expected number of crime with manually generated strategy is close to the real number of crimes, which indicates game abstraction model captures the feature of criminals and provide good estimation of the real number of crimes. In addition, strategy generated using the game abstraction is projected to outperform the manually generated strategy significantly. This shows the effectiveness of our proposed patrol strategy as compared to the current patrol strategy.

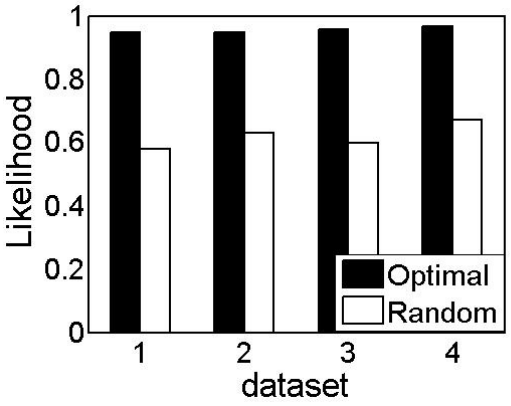


Figure 5.16: Likelihood (city)

Runtime: Finally, we break down the total runtime of the game abstraction framework in the city problem layer by layer and show it in Figure 5.17. The x-axis is the index of the layer, which goes from the original layer (Layer 1) to the top layer (Layer 5). The y-axis is the total runtime of the propagation, learning and planning algorithm in that layer. As can be seen, the runtime increases as the layer index decreases except for Layer 1. This is because in greedy layer generation, for the fifth layer the number of targets is 5, and for the fourth layer it is 5^2, for third layer it is 5^3, for the second layer it is 5^4 but for the first layer it is only 900. Therefore, the number of targets within each *aggregated target* in layer two is less than $3 < n = 5$. Therefore, the runtime in layer 1 is faster. However, the total runtime of the whole process is less than an hour in each data set. Therefore, the game abstraction framework can be extended to large scale problems with reasonable runtime performance.

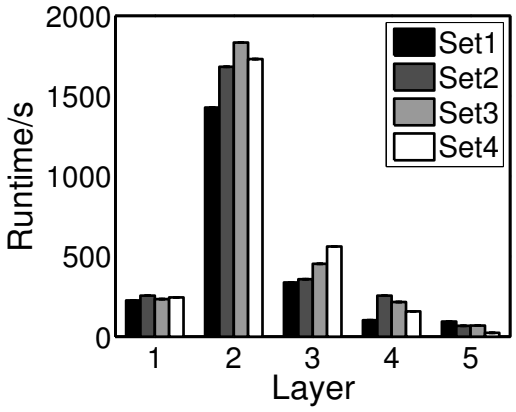


Figure 5.17: Runtime

Chapter 6

Example of learnt DBN for USC problem

6.1 USC background

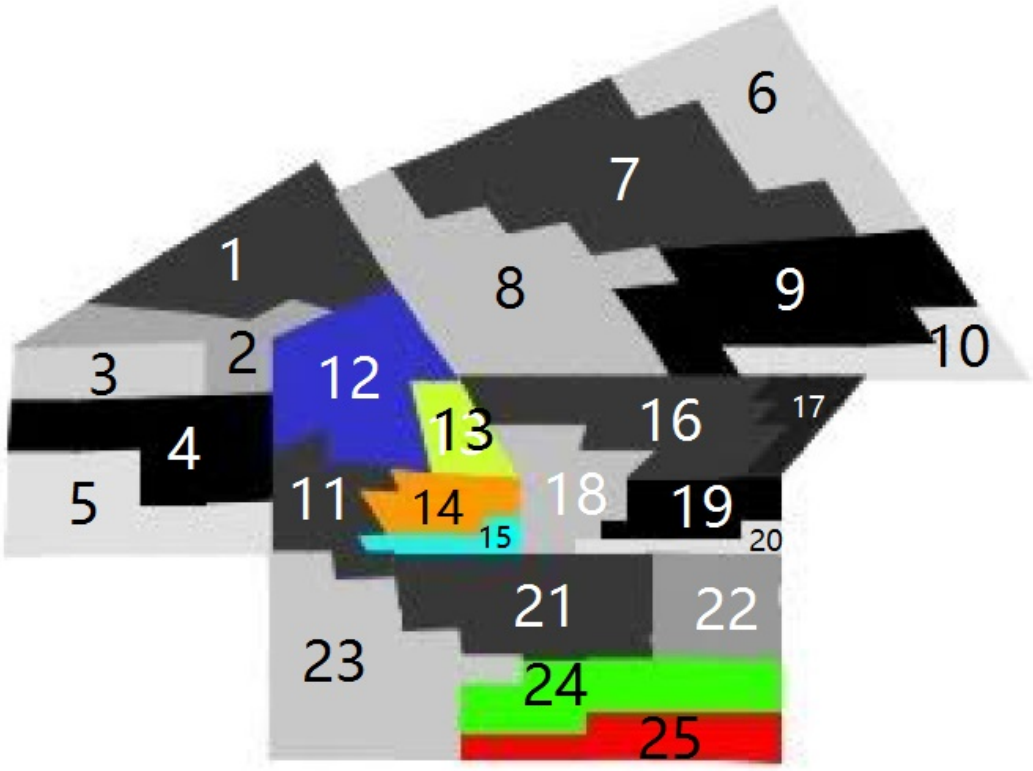


Figure 6.1: USC campus map

In order to provide more detailed patrol strategies for defender against opportunistic criminals in USC areas, we divide the whole campus into 25 targets instead of 5 targets. We want to learn the criminals' behavior on these 25 targets and recommend patrol strategies, which is shown in Figure 6.1 shows the 25 targets segmentation. There is geometric distortion in the map while the relative relation of targets keeps the same. In Figure 6.1, we further divide these 5 targets into 25 targets. 6 targets are painted with colors. Purple, yellow, orange and blue targets are in the campus region while green and red targets are in region B, which is the Exposition Park. We will focus on these five targets later in crime matrix.

As the result of the abstract game, we are able to generate the criminals' crime matrix as well as movement matrix for these 25 targets in Figure 6.1 for USC problem. Recall that as introduced in DBN model in Figure 5.4, crime matrix contains probabilities of criminals committing a crime at each target with the defender's patrol strategy. The movement matrix contains probabilities of criminals moving to a targets at next time step with the defender's patrol strategy.

6.2 Crime Matrix

As stated before, the crime matrix is factorized and each element of the matrix represents the probability $P(Y_{i,t}|D_{i,t}, X_{i,t})$, where i is the index of the target, t is the time step. $Y_{i,t}$ is the number of crimes at target i at step t; $D_{i,t}$ is the number of defenders at target i at step t and $X_{i,t}$ is the number of criminals at target i at step t. Therefore, $P(Y_{i,t}|D_{i,t}, X_{i,t})$ is the probability that there will be $Y_{i,t}$ crimes given $D_{i,t}$ defenders and $X_{i,t}$ criminals.

In binary cases that we are considering, $Y_{i,t} \in \{0, 1\}$, indicating there will be no crime or there will be at least one crime; $D_{i,t} \in \{1, 2\}$, indicating there will be at most one defenders or there will be at least two defenders and $X_{i,t} \in \{0, 1\}$, indicating there will be no criminals or there will be at least one criminal. Therefore, the crime matrix should be $2 \times 2 \times 2 \times N$, where N is the number of targets in the problem. However, there are two facts to help reduce the size of the crime matrix: 1, $P(Y_{i,t} = 1|D_{i,t}, X_{i,t}) + P(Y_{i,t} = 0|D_{i,t}, X_{i,t}) = 1$, meaning that the sum of probability of having a crime and not having a crime should be 1 for each targets. Therefore, we only need to show $P(Y_{i,t} = 1|D_{i,t}, X_{i,t})$ and $P(Y_{i,t} = 0|D_{i,t}, X_{i,t})$ is just the compensation. Therefore, the sime of the crime matrix is trim down to $2 \times 2 \times N$; 2, moreover, $P(Y_{i,t} = 1|D_{i,t}, X_{i,t} = 0) = 0$, which means that if there is no criminal at target i at step t, there will be no crime at target i at target t. This is true for all i and therefore we omit this part in the crime matrix. By applying these two facts, the size of the crime matrix is trim down to $2 \times N$ where only $P(Y_{i,t} = 1|D_{i,t}, X_{i,t} = 1)$ are shown.

Figure 6.2 shows the crime matrix for $N = 25$ targets problem. We split this 2×25 crime matrix into 4 lines. $D = 1$ represents there is at most one defender protecting the target while $D = 2$ represents that at least two defenders are protecting the target. There are two conclusions we can draw from the crime matrix: First, a general conclusion is that for each target, the probability for criminals to commit a crime when the defender is protecting the target is lower than the probability that the criminals will commit a crime when the defender is not at that target. Clearly this shows a difference due to defender presence, that is, the defender's appearance has negative effect on criminal's probability of committing a crime. The more defenders are at a target, the less likely the criminal

N	1	2	3	4	5	6
D=1	0.5584	0.5823	0.6739	0.4428	0.5345	0.2384
D=2	0.3083	0.3242	0.4157	0.2209	0.1814	0.0911
N	7	8	9	10	11	12
D=1	0.1750	0.2067	0.1953	0.2181	0.7481	1
D=2	0.0627	0.0769	0.0723	0.0815	0.1692	0.2287
N	13	14	15	16	17	18
D=1	1	0.4961	1	0.2439	0.2303	0.2905
D=2	0.1438	0.1946	0.2542	0.1377	0.1299	0.1694
19	20	21	22	23	24	25
0.1702	0.2168	0.1322	0.1414	0.0432	0.123	0.2212
0.090	0.1221	0.0691	0.0742	0.0205	0.064	0.1176

Figure 6.2: Crime Matrix

commit a crime at this target. Secondly, the impact of defenders at different targets are different. For example, the defender can reduce crime dramatically in target $12, 13, 14$ and 15, which are the purple, green, orange and blue targets respectively in Figure 6.1. At the same time, the difference of crime probability with different number of defenders is smaller in target 24 and 25, which are the greed and red targets respectively. In order to explain such phenomenon, we form the following hypothesis: As stated before, $12, 13$ and 15 are in campus region, there will be more crime opportunities due to the large amount of students. At the same time, target 24 and 25 are in exposition park that usually do not have too many crime opportunities. Therefore, the crime probability are higher in $12, 13, 15$ than that in 24 and 25. At the same time, the defender will be more

effective in campus because i) the campus region is smaller; ii) the transportation is easier and iii) there are facilities such as cameras to assist them. Therefore, the probability of committing a crime drops largest in these targets when more defenders are protecting. In target 24 and 25, i) the criminal are less active; ii) the region is large for patrol and iii) there are no facilities to help the defenders. Therefore the effect of increasing defenders are smaller in reducing crime probabilities.

6.3 Movement Matrix

N	1	2	3	4	5	6
D=1	0.4275	0.05309	0.06113	0.0514	0.05906	0.04642
D=2	0.4050	0.0555	0.06391	0.0537	0.06175	0.04853
N	7	8	9	10	11	12
D=1	0.04548	0.04898	0.05068	0.05435	0.09714	0.13708
D=2	0.04755	0.05120	0.05298	0.05682	0.10156	0.14331
N	13	14	15	16	17	18
D=1	0.15659	0.11031	0.20293	0.11683	0.12895	0.15513
D=2	0.16371	0.11532	0.21216	0.12214	0.13481	0.16219
19	20	21	22	23	24	25
0.16014	0.19636	0.21964	0.27698	0.33558	0.52301	0.912
0.16741	0.20529	0.22963	0.28957	0.35083	0.54679	0.963

Figure 6.3: Movement Matrix

For movement matrix, it describes how the criminals move with respect to different allocation strategies of defender. That is, the element in the movement matrix represents the probability $P(X_{i,t+1}|X_{j,t}, D_{j,t})$. $X_{i,t+1}$ is the number of criminals at target i at step t+1; $X_{j,t}$ is the number of criminals at target j at step t and $D_{j,t}$ is the number of defenders at target j at step t. Since the criminal can move from any target to any target, therefore, the size of the movement matrix is $N \times 2 \times N \times 2 \times 2$ since we need to consider all the targets at step i and all the targets at step j. Again, we can reduce the size of the movement matrix by making use of 2 facts: 1, $P(X_{i,t+1} = 1|X_{j,t}, D_{j,t}) + P(X_{i,t+1} = 0|X_{j,t}, D_{j,t}) = 1$ which indicates the probability of criminal at target i at step $t + 1$ and the probability of criminal not at target i at step $t + 1$ are sum up to 1. Therefore, we only show $P(X_{i,t+1} = 1|X_{j,t}, D_{j,t})$ in the movement matrix; 2, $P(X_{i,t+1} = 1|X_{j,t} = 0, D_{j,t}) = 0$ which indicates if there is no criminal at target j at step t, then there will be no criminal transition from target j to any other target i. Therefore, the size of the movement matrix is reduced to $N \times N \times 2$.

Figure 6.3 shows the part of the movement matrix where $j = 1$, which is the transition probability originated at target 1 at step t to all the targets at step $t + 1$, $P(X_{i,t+1} = 1|X_{1,t} = 1, D_{1,t})$. At can be seen in Figure 6.3, the transition probability to target 1 when there is 1 defender $P(X_{1,t+1} = 1|X_{1,t} = 1, D_{1,t} = 1) = 0.4275$ is higher than the probability when there is 2 defenders $P(X_{1,t+1} = 1|X_{1,t} = 1, D_{1,t} = 2) = 0.4050$. This indicates that the criminals are less likely to stay at the same target when more defenders are showing up at this target. For all other targets $i \in 2, ..., 25$, $P(X_{i,t+1} = 1|X_{1,t} = 1, D_{1,t} = 1) < P(X_{i,t+1} = 1|X_{1,t} = 1, D_{1,t} = 2)$, which indicates that the more defenders are at target 1, the more likely that the criminal will transit to another target than 1. As

we can summarized from the movement matrix originated from target 1, the defender's appearance at target 1 disperses the criminals. That is, the criminals are less likely to stay at the target 1 if the defender is at this target. The trends are the same for all other 24 targets from the other part of the movement matrix, which is omitted because of the space limit.

Chapter 7

Conclusion and Future Work

7.1 Summary

My thesis first introduces the Opportunistic Security Game, a new computational framework to address opportunistic crime, opening the door for further research on this topic. We propose a new exact algorithm, EOSG, to compute defender resource allocation strategies, and an approximate algorithm, COPS, to speed up defender allocation to real-world scale scenarios. Our experimental results show that the OSG strategy outperforms baseline strategies with different types of criminals. We also show that COPS is more efficient than EOSG in solving real-world scale problems. Given our experimental results, COPS is being evaluated in the Los Angeles Metro system. In introducing OSG, my thesis has added to the class of important security-focused game-theoretic frameworks in the literature, opening the door to a new set of research challenges for the community of researchers focused on game theory for security.

Secondly, my thesis introduces a novel framework to learn and plan against adaptive opportunistic criminals using real data. First, we model the interaction between officers

and adaptive opportunistic criminals as a DBN. Next, we propose a sequence of modifications to the basic DBN resulting in a compact model that enables better learning accuracy and running time. Finally, we present an iterative learning and planning mechanism with two planning algorithm to keep pace with adaptive opportunistic criminals. Experimental validation with real data supports our choice of model and assumptions. Further, our modeling assumptions were informed by inputs from our collaborators in the DPS at USC. These promising results have opened up the possibility of deploying our method in USC. My thesis has further opened up the integration of opportunistic crime security games [59] with machine learning.

Finally, my thesis introduces a novel game abstraction framework to learn and plan against opportunistic criminals in large-scale urban areas. First, we model the layer-generating process as a districting problem and propose a MILP based technique to solve the problem. Next, we propose a planning algorithm that outputs randomized strategies. Finally, we use a heuristic propagation model to handle the problem with limited data. Experiments with real data in two urban settings shows that our framework can handle large scale urban problems that previous state-of-the-art techniques fail to scale up to. Further, our approach provides high crime prediction accuracy and the strategy generated from our framework is projected to significantly reduce crime compared to current police strategy.

In summary, in my thesis I propose a novel framework to learn and plan against adaptive opportunistic criminals using real world data. This general framework can be applied to opportunistic crime problem in any urban settings such as USC campus, city of Nashville and Los Angeles Metro System. Moreover, this framework can also be applied

to opportunistic crime problem in other domains. Two examples are the wildlife poaching problem and the cyber security problem. There are a lot of endangered species such as elephants and rhinos that are the target of poachers. Patrol units are sent in the field to prevent the wildlife poaching. Such wildlife poaching problem can be modeled as an Opportunistic Crime Security Game. In this game, the opportunistic criminals are the poachers who put snares to capture wild animals. The defender are the patrol units who conduct foot patrol to destroy these snares. The interesting feature in this problem is that the defender can only get partial observation of crime activity due to the limited patrol resource and huge area to patrol. Therefore, Partially Observable Markov Decision Processes (POMDP) should be considered in this Opportunistic Criminal Security Game for wildlife poaching problem. At the same time, when designing the patrol strategy in the poaching problem, we need to consider the exploration-exploitation trade-off. That is, at each step, the patrol units need to decide whether to protect the targets that they have visited before to exploit the criminal's behavior at those targets or explore the unknown targets that they have never visited before. The second example is the cyber security problem. In this problem, there are a cluster of machines/users in a computer network. The opportunistic criminals try to hack and attack the vulnerable machine in this network. At the same time, the defender analyzes the outside visit to detect the attacks. It can also be modeled as an Opportunistic Crime Security Game. Given the limited resources, the defenders cannot detect every visit so she has to decide the subset of machines she have to protect at each time. Since the criminals frequently attack the network and may change their behavior rapidly from the result the observed, fast online learning and planning algorithms are needed in this problem.

Even though the Opportunistic Security Game has various extensions, it has following four limitations: First, the Opportunistic Security Game framework assumes the criminals are opportunistic and not always make the decision that maximize their utilities. Such setting may not be efficient when we deal with attackers that are rational. For example, in terrorist attacks, the terrorists will conduct long time surveillance and make careful plan before executing the attack. Therefore, the Opportunistic Security Game may not be a good model for this kind of problems. Second, this framework requires adequate amount of data in order to learn the Dynamic Bayesian Network. By our experience, the amount of available data should be at least ten times more than the unknown variables. Since the scale of unknown variables is $O(N^2)$, where N is the number of targets in the problem, the amount of data that is needed grows rapidly with N. Therefore, for large scale problems, the available data points may not be adequate and the learned criminal model may be inefficient. The third limitation of this framework is the runtime of the algorithm. Even after all the accelerating approaches, the runtime grows exponentially to the scale of the problem and the number of officers. Therefore, it may take days to generate patrol strategy in large scale problem while the officers may want to know their patrol schedule for next hour or next shift. How to generate patrol strategy in real time fashion remains to be a problem for this model. The final limitation of this model is the real world execution uncertainty. In our paper, we assume that the patrol officers can always perfectly execute the patrol strategy. However, this might not be true in real world scenarios. Police officers may have execution uncertainties such as sickness or personal emergency. Sometimes they may not follow the patrol schedule we generated for them simply because they don't like it. Therefore, how can we address

these execution uncertainties remain to be a problem. In order to solve this problem, we need to implement our algorithm and get feedback from officers in the field.

There are four implications from this framework: i) it is efficient to model the whole group of opportunistic criminals instead of each individual. Even though criminals have different background and may behave differently, our model considers them to be homogeneous. Results show that such setting helps better predict crimes, which implies that the criminals behave similarly when committing crimes regardless of their background; ii) as mentioned above, instead of profiling different types of people, our model focuses on the general behavior patterns of the criminals. That is, we do not divide them by any properties such as gender or race. Under this setting we can generate accurate criminal model without revealing the personal information. However, we are aware of research on explainable machine learning and knowledge discovery systems [33, 47]. In these works, researchers are trying to figure out the important factors that affect the system's predicting ability. However, we need to be extremely careful if we want to introduce such system in our domain since personal information and human rights must be considered and protected; iii) the behavior pattern of opportunistic criminals can be mostly learned with crime and patrol historical data. While there are a lot of factors that may affect the criminal's decision of committing a crime such as environment, weather and so on, our result shows that only considering the interaction between crime and patrol officer's allocation can achieve high likelihood, which indicates patrol is a major factor that affects criminals' behavior; iv) defenders' mixed strategy is unpredictable for opportunistic criminals and therefore more efficient against such kind of criminals. As shown in our experiments, when defenders follow the strategy that is manually generated by domain

experts, criminals can easily figure out the pattern and commit more crimes. At the same time, when defenders apply the mixed strategy that is provided by the Opportunistic Security Game model, it is harder for criminals to find out the pattern. This results in less crime that they commit.

7.2 Future Work

My current work has provided models and algorithms to handle opportunistic criminals in urban areas. There are three possible directions to explore.

7.2.1 Combine two approaches

A straightforward idea is to combine Opportunistic Security Games with the machine learning based approach. The advantage of machine learning approach is that it learns criminals behavior model directly from real world data and it is a parameter free model, meaning that we do not need to make prior assumptions on criminals behavior. However, it requires adequate amount of data points to estimate such model. OSG, on the other hands, only require little data to build criminals behavior, but set constraints for criminal's behavior such as quantal biased random movement. Even though these constraints are justified in game theory and computational criminology, we still need to verify whether they are suitable for opportunistic criminals. Therefore, by combining these two approaches, we want to lean a game-theoretic model that contains enough parameters to describe opportunistic criminals.

7.2.2 Improve model and algorithm

Even though COPS algorithm has already accelerated the computing, the runtime still grows exponentially to the scale of the problem and the number of officers. A nature question is whether there is a faster algorithm for recommending patrol strategies. After doing a survey on existing literature, I haven't found an fast approach for my models. Therefore, I am now working on developing fast algorithms for generating patrol strategy. Moving forward, I also expect to generate patrol schedule in a more detailed level. Current approaches consider the urban area in an abstract level, ignoring the internal structure for each target. In reality, such internal structure, such as the platform and parking lots of a train station and the library on campus, is crucial for designing patrol. Besides, there are different types of patrol officers, who has different duties and different impacts on the crime. Finally, we need to consider the types of crimes, which are committed by criminals with different behavior patterns. In order to handle these situations, more sophisticated approaches are needed.

7.2.3 Real-world Implementation

Finally, these algorithms will eventually be implemented in real urban area for evaluating and improving. I will implement these algorithms to schedule patrol strategies on campus with the help of the Department of Public Safety in University of Southern California. To fill the gap between simulation and implementation, I need to consider practical constraints such as the appearance of events, e.g. football games, and emergency. Meanwhile, the defender's patrol preference is also an important factor. For example, within the same

target area, some officers spend more time near library while others patrol gym more frequently. Also, building a potable device or software for the officers is a non-trivial task. I have submitted the initial model to the Demo session of AAMAS 2015 and I will keep improving it.

References

[1] Srinivas M Aji and Robert J McEliece. The generalized distributive law. *Information Theory, IEEE Transactions on*, 46(2):325–343, 2000.

[2] Anjon Basak and Christopher Kiekintveld. Abstraction using analysis of subgames. In *IJCAI Workshop on Algorithmic Game Theory*, 2015.

[3] Nicola Basilico and Nicola Gatti. Automated abstractions for patrolling security games. In *AAAI*, 2011.

[4] Nicola Basilico and Nicola Gatti. Strategic guard placement for optimal response toalarms in security games. In *Proceedings of the 2014 international conference on Autonomous agents and multi-agent systems*, pages 1481–1482. International Foundation for Autonomous Agents and Multiagent Systems, 2014.

[5] Nicola Basilico, Nicola Gatti, and Francesco Amigoni. Leader-follower strategies for robotic patrolling in environments with arbitrary topologies. In *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems-Volume 1*, pages 57–64. International Foundation for Autonomous Agents and Multiagent Systems, 2009.

[6] Nicola Basilico, Nicola Gatti, and Francesco Amigoni. Patrolling security games: Definition and algorithms for solving large instances with single patroller and single intruder. *Artificial Intelligence*, 184:78–123, 2012.

[7] Nicola Basilico, Nicola Gatti, Thomas Rossi, Sofia Ceppi, and Francesco Amigoni. Extending algorithms for mobile robot patrolling in the presence of adversaries to more realistic settings. In *Proceedings of the 2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology-Volume 02*, pages 557–564. IEEE Computer Society, 2009.

[8] Dimitris Bertsimas and John N Tsitsiklis. *Introduction to linear optimization*, volume 6. Athena Scientific Belmont, MA, 1997.

[9] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[10] Avrim Blum, Nika Haghtalab, and Ariel D Procaccia. Learning optimal commitment to overcome insecurity. In *Proceedings of the 28th Annual Conference on Neural Information Processing Systems (NIPS). Forthcoming*, 2014.

[11] Alselm Blumer, Andrzej Ehrenfeucht, David Haussler, and Manfred K. Warmuth. Occam's razor. *Inf. Process. Lett.*, 24(6):377–380, April 1987.

[12] Xavier Boyen and Daphne Koller. Tractable inference for complex stochastic processes. In *Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence*, pages 33–42. Morgan Kaufmann Publishers Inc., 1998.

[13] P Jeffrey Brantingham and George Tita. Offender mobility and crime pattern formation from first principles. *Artificial crime analysis systems*, 2008.

[14] Patricia Brantingham and Paul Brantingham. Criminality of place. *European Journal on Criminal Policy and Research*, 3(3):5–26, 1995.

[15] Victor Bucarey, Fernando Ordóñez, and Enrique Bassaletti. Shape and balance in police districting. In *Applications of Location Analysis*, pages 329–347. Springer, 2015.

[16] Hsinchun Chen, Wingyan Chung, Jennifer Jie Xu, Gang Wang, Yi Qin, and Michael Chau. Crime data mining: a general framework and some examples. *Computer*, 37(4):50–56, 2004.

[17] Andrew Clark, Quanyan Zhu, Radha Poovendran, and Tamer Başar. Deceptive routing in relay networks. In *Decision and Game Theory for Security*, pages 171–185. Springer, 2012.

[18] Vincent Conitzer and Tuomas Sandholm. A technique for reducing normal-form games to compute a nash equilibrium. In *Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, pages 537–544. ACM, 2006.

[19] Derek B Cornish and Ronald V Clarke. Understanding crime displacement: An application of rational choice theory. *Criminology*, 1987.

[20] Jeroen S De Bruin, Tim K Cocx, Walter Kosters, Jeroen FJ Laros, Joost N Kok, et al. Data mining approaches to criminal career analysis. In *Data Mining, 2006. ICDM'06. Sixth International Conference on*, pages 171–177. IEEE, 2006.

[21] Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 1–38, 1977.

[22] Fei Fang, Peter Stone, and Milind Tambe. When security games go green: Designing defender strategies to prevent poaching and illegal fishing. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2015.

[23] Andrew Gilpin and Tuomas Sandholm. Better automated abstraction techniques for imperfect information games, with application to texas hold'em poker. In *Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems*, page 192. ACM, 2007.

[24] Andrew Gilpin and Tuomas Sandholm. Lossless abstraction of imperfect information games. *Journal of the ACM (JACM)*, 54(5):25, 2007.

[25] Andrew Gilpin, Tuomas Sandholm, and Troels Bjerre Sørensen. A heads-up no-limit texas hold'em poker player: discretized betting models and automatically generated equilibrium-finding programs. In *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems-Volume 2*, pages 911–918. International Foundation for Autonomous Agents and Multiagent Systems, 2008.

[26] Joao P Hespanha, Maria Prandini, and Shankar Sastry. Probabilistic pursuit-evasion games: A one-step nash approach. In *Decision and Control, 2000. Proceedings of the 39th IEEE Conference on*, volume 3, pages 2272–2277. IEEE, 2000.

[27] Christian Ivaha, Hasan Al-Madfai, Gary Higgs, and J Andrew Ware. The dynamic spatial disaggregation approach: A spatio-temporal modelling of crime. In *World congress on engineering*, pages 961–966, 2007.

[28] Albert Xin Jiang, Zhengyu Yin, Chao Zhang, Milind Tambe, and Sarit Kraus. Game-theoretic randomization for security patrolling with dynamic execution uncertainty. In *AAMAS*, 2013.

[29] Jörg Kalcsics, Stefan Nickel, and Michael Schröder. Towards a unified territorial design approachapplications, algorithms and gis integration. *Top*, 13(1):1–56, 2005.

[30] Oded Kariv and S Louis Hakimi. An algorithmic approach to network location problems. ii: The p-medians. *SIAM Journal on Applied Mathematics*, 37(3):539–560, 1979.

[31] Leslie W Kennedy, Joel M Caplan, and Eric Piza. Risk clusters, hotspots, and spatial intelligence: risk terrain modeling as an algorithm for police resource allocation strategies. *Journal of Quantitative Criminology*, 27(3):339–362, 2011.

[32] Joshua Letchford, Liam MacDermed, Vincent Conitzer, Ronald Parr, and Charles L Isbell. Computing optimal strategies to commit to in stochastic games. In *AAAI*, 2012.

[33] Benjamin Letham, Cynthia Rudin, Tyler H McCormick, David Madigan, et al. Interpretable classifiers using rules and bayesian analysis: Building a better stroke prediction model. *The Annals of Applied Statistics*, 9(3):1350–1371, 2015.

[34] Ned Levin, N Levine, et al. Crimestat iii-a spatial statistics program for the analysis of crime incident locations. *US Department of Justice, Houston*, 2004.

[35] Anastasia Loukaitou-Sideris, Robin Liggett, and Hiroyuki Iseki. The geography of transit crime documentation and evaluation of crime incidence on and around the green line stations in los angeles. *JPER*, 2002.

[36] Anuj Malik, Ross Maciejewski, Sherry Towers, Sean McCullough, and David S Ebert. Proactive spatiotemporal resource allocation and predictive visual analytics for community policing and law enforcement. *Visualization and Computer Graphics, IEEE Transactions on*, 20(12):1863–1872, 2014.

[37] R. D. McKelvey and T. R. Palfrey. Quantal Response Equilibria for Normal Form Games. *Games and Economic Behavior*, 10(1):6–38, 1995.

[38] George O Mohler, Martin B Short, P Jeffrey Brantingham, Frederic Paik Schoenberg, and George E Tita. Self-exciting point process modeling of crime. *Journal of the American Statistical Association*, 2012.

[39] Alan T Murray, Ingrid McGuffog, John S Western, and Patrick Mullins. Exploratory spatial data analysis techniques for examining urban crime implications for evaluating treatment. *British Journal of criminology*, 41(2):309–329, 2001.

[40] Shyam Varan Nath. Crime pattern detection using data mining. In *Web Intelligence and Intelligent Agent Technology Workshops, 2006. WI-IAT 2006 Workshops. 2006 IEEE/WIC/ACM International Conference on*, pages 41–44. IEEE, 2006.

[41] Thanh H Nguyen, Francesco M Delle Fave, Debarun Kar, Aravind S Lakshminarayanan, Amulya Yadav, Milind Tambe, et al. Making the most of our regrets: Regret-based solutions to handle payoff uncertainty and elicitation in green security games. In *Decision and Game Theory for Security*, pages 170–191. Springer, 2015.

[42] Giles Oatley, Brian Ewart, and John Zeleznikow. Decision support systems for police: Lessons from the application of data mining techniques to soft forensic evidence. *Artificial Intelligence and Law*, 14(1-2):35–100, 2006.

[43] James Pita, Manish Jain, Fernando Ordóñez, Milind Tambe, Sarit Kraus, and Reuma Magori-Cohen. Effective solutions for real-world stackelberg games: When agents must deal with human uncertainties. In *AAMAS*, 2009.

[44] Aaron Plavnick. The fundamental theorem of markov chains. University of Chicago VIGRE REU, 2008.

[45] Y Qian, X Jiang, W Haskell, and M Tambe. Online planning for optimal protector strategies in resource conservation games. AAMAS, 2014.

[46] Jerry H Ratcliffe. A temporal constraint theory to explain opportunity-based spatial offending patterns. *Journal of Research in Crime and Delinquency*, 2006.

[47] Cynthia Rudin, David Waltz, Roger N Anderson, Albert Boulanger, Ansaf Salleb-Aouissi, Maggie Chow, Haimonti Dutta, Philip N Gross, Bert Huang, Steve Ierome, et al. Machine learning for the new york city power grid. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 34(2):328–345, 2012.

[48] Tuomas Sandholm and Satinder Singh. Lossy stochastic game abstraction with bounds. In *Proceedings of the 13th ACM Conference on Electronic Commerce*, pages 880–897. ACM, 2012.

[49] Kelly E See, Craig R Fox, and Yuval S Rottenstreich. Between ignorance and truth: Partition dependence and learning in judgment under uncertainty. *Journal of Experimental Psychology*, 2006.

[50] Hanif D Sherali and Frederick L Nordai. Np-hard, capacitated, balanced p-median problems on a chain graph with a continuum of link demands. *Mathematics of Operations Research*, 13(1):32–49, 1988.

[51] Martin B Short, Maria R D'ORSOGNA, Virginia B Pasour, George E Tita, Paul J Brantingham, Andrea L Bertozzi, and Lincoln B Chayes. A statistical model of criminal behavior. *Mathematical Models and Methods in Applied Sciences*, 18(supp01):1249–1267, 2008.

[52] Paul W Speer, Dennis M Gorman, Erich W Labouvie, and Mark J Ontkush. Violent crime and alcohol availability: relationships in an urban community. *Journal of public health policy*, pages 303–318, 1998.

[53] Milind Tambe. *Security and game theory: algorithms, deployed systems, lessons learned*. Cambridge University Press, 2011.

[54] Traci L Toomey, Darin J Erickson, Bradley P Carlin, Harrison S Quick, Eileen M Harwood, Kathleen M Lenk, and Alexandra M Ecklund. Is the density of alcohol establishments related to nonviolent crime? *Journal of studies on alcohol and drugs*, 73(1):21–25, 2012.

[55] Bernhard Von Stengel and Shmuel Zamir. Leadership with commitment to mixed strategies. 2004.

[56] Yevgeniy Vorobeychik, Bo An, and Milind Tambe. Adversarial patrolling games. In *In AAAI Spring Symposium on Security, Sustainability, and Health.*, 2012.

[57] Yevgeniy Vorobeychik and Satinder P Singh. Computing stackelberg equilibria in discounted stochastic games. In *AAAI*, 2012.

[58] Zhengyu Yin, Albert Xin Jiang, Milind Tambe, Christopher Kiekintveld, Kevin Leyton-Brown, Tuomas Sandholm, and John P Sullivan. Trusts: Scheduling randomized patrols for fare inspection in transit systems using game theory. *AI Magazine*, 33(4):59, 2012.

[59] Chao Zhang, Albert Xin Jiang, Martin B Short, P Jeffrey Brantingham, and Milind Tambe. Defending against opportunistic criminals: New game-theoretic frameworks and algorithms. In *Decision and Game Theory for Security*, pages 3–22. Springer, 2014.

[60] Chao Zhang, Arunesh Sinha, and Milind Tambe. Keeping pace with criminals: Designing patrol allocation against adaptive opportunistic criminals. In *AAMAS 2015*.

[61] Quanyan Zhu, Andrew Clark, Radha Poovendran, and Tamer Basar. Deceptive routing games. In *Decision and Control (CDC), 2012 IEEE 51st Annual Conference on*, pages 2704–2711. IEEE, 2012.

[62] Joseph R Zipkin, Martin B Short, and Andrea L Bertozzi. Cops on the dots in a mathematical model of urban crime and police response. 2013.