games

MDPI

Article

Keeping Pace with Criminals: An Extended Study of Designing Patrol Allocation against Adaptive Opportunistic Criminals

Chao Zhang [1,*], Shahrzad Gholami [1], Debarun Kar [1], Arunesh Sinha [1], Manish Jain [2], Ripple Goyal [2] and Milind Tambe [1]

[1] Department of Computer Science, University of Southern California, Los Angeles, CA 90089, USA; sgholami@usc.edu (S.G.); dkar@usc.edu (D.K.); aruneshsinha@gmail.com (A.S.); tambe@usc.edu (M.T.)

[2] Armorway. Inc., Los Angeles, CA 90291, USA; manish@armorway.com (M.J.); ripple@armorway.com (R.G.)

* Correspondence: zhan661@usc.edu; Tel.: +1-213-880-8925

Academic Editors: Karl Tuyls and Simon Parsons

Received: 13 March 2016; Accepted: 19 June 2016; Published: 27 June 2016

Abstract: Game theoretic approaches have recently been used to model the deterrence effect of patrol officers' assignments on opportunistic crimes in urban areas. One major challenge in this domain is modeling the behavior of opportunistic criminals. Compared to strategic attackers (such as terrorists) who execute a well-laid out plan, opportunistic criminals are less strategic in planning attacks and more flexible in executing well-laid plans based on their knowledge of patrol officers' assignments. In this paper, we aim to design an optimal police patrolling strategy against opportunistic criminals in urban areas. Our approach is comprised by two major parts: learning a model of the opportunistic criminal (and how he or she responds to patrols) and then planning optimal patrols against this learned model. The planning part, by using information about how criminals responds to patrols, takes into account the strategic game interaction between the police and criminals. In more detail, first, we propose two categories of models for modeling opportunistic crimes. The first category of models learns the relationship between defender strategy and crime distribution as a Markov chain. The second category of models represents the interaction of criminals and patrol officers as a Dynamic Bayesian Network (DBN) with the number of criminals as the unobserved hidden states. To this end, we: (i) apply standard algorithms, such as Expectation Maximization (EM), to learn the parameters of the DBN; (ii) modify the DBN representation that allows for a compact representation of the model, resulting in better learning accuracy and the increased speed of learning of the EM algorithm when used for the modified DBN. These modifications exploit the structure of the problem and use independence assumptions to factorize the large joint probability distributions. Next, we propose an iterative learning and planning mechanism that periodically updates the adversary model. We demonstrate the efficiency of our learning algorithms by applying them to a real dataset of criminal activity obtained from the police department of the University of Southern California (USC) situated in Los Angeles, CA, USA. We project a significant reduction in crime rate using our planning strategy as compared to the actual strategy deployed by the police department. We also demonstrate the improvement in crime prevention in simulation when we use our iterative planning and learning mechanism when compared to just learning once and planning. Finally, we introduce a web-based software for recommending patrol strategies, which is currently deployed at USC. In the near future, our learning and planning algorithm is planned to be integrated with this software. This work was done in collaboration with the police department of USC.

Keywords: security games; optimization; game theory

1. Introduction

Urban crime plagues every city across the world. 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 [1,2]. In order to deter such crimes, police officers conduct patrols in an attempt to prevent crime. However, by observing the actual presence of patrol units, criminals can adapt their strategy by seeking crime opportunity in less effectively-patrolled locations. The problem of where and how much to patrol is, therefore, important.

Previously, two broad approaches have been used to attempt to solve this problem. The first approach is to manually determine patrol schedules through use of human planners. This approach is used in various police departments, including the University of Southern California (USC). However, it has been demonstrated in other related scenarios, such as the protection of airport terminals [3] and ships in ports [4], that manually planning patrols is not only time consuming, but is also highly ineffective. The second approach is to use automated planners to plan patrols against urban crime. This approach has either focused on modeling the criminal explicitly [1,2] (rational, bounded rational, limited surveillance, etc.) in a game model or learning the adversary behavior using machine learning [5]. However, the proposed mathematical models of criminal behavior have not been validated with real data. Furthermore, prior machine learning approaches have only focused on the adversary actions, ignoring their adaptation to the defenders' actions [5].

Hence, in this paper, we tackle the problem of generating patrol strategies against opportunistic criminals. We propose the following novel approach: aim to progressively learn criminal behavior using real data. We do so by using two approaches to model the interaction between the criminal and patrol officers: the *Markov chain model* (MCM) and the *Dynamic Bayesian Network (DBN) model* (DBNM). More specifically, MCM directly relates crime to observed data, such as past crime and patrols. This model category follows concepts from prior literature; e.g., capturing crime phenomena, such as "crime predicts crime".

DBNM represents the interaction between the criminal and patrol officers as a Dynamic Bayesian Network (DBN) with the number of criminals as the unknown (or latent) state. 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. Therefore, we propose a sequence of modifications of the initial DBN model resulting in a compact representation of the model. This leads to better learning accuracy and increased speed of learning of the EM algorithm when used for the compact model. This sequence of modifications involves marginalizing states in the DBN using approximation techniques from the Boyen–Koller algorithm [6] and exploiting the 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 next contributions 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's behavior would change as he or she observes and reacts to the deployment of a new strategy. Hence, the optimal strategy with respect to the learned behavior may not be effective for a long time, as the adversary behavior may change. 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 a more effective officer strategy that benefits from adaptation.

Next, as part of our collaboration with the police department of USC, we obtained criminal activity and patrol data covering a range of three years. This collaboration not only helped us validate our learning approach, but it also provided insights about the sequence of modifications that could be made for Markov chain models, as well as 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 Section 6). 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.

Finally, we build a web-based software that is a patrol schedule recommendation system. In addition, our system collects and analyzes crime reports and resource (security camera, emergency supplies, etc.) data, presenting them in various user-friendly forms. The software is currently deployed for use by the USC police department around their Los Angeles, CA, campus. In future, there are plans to incorporate our learning and planning approach with this software.

2. Related Work

We categorize the related work into five overarching areas. First, recent research has made inroads in applying machine learning and data mining in the domain of criminology to analyze crime patterns and support police in making decisions. A general framework for crime data mining is introduced in [5]. In [7], data mining is used to model crime detection problems and cluster crime patterns; in [8], data mining approaches are applied in criminal career analysis; in [9], 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.

The second line of work we compare with is Pursuit-Evasion Games (PEG). PEG models a pursuer(s) attempting to capture an evader, often where their movement is based on a graph [10]. However, in common settings of pursuit evasion games, an evader's goal is to avoid capture, not to seek opportunities to commit crimes, while a pursuer's goal is to capture the evader, not to deter the criminal. Thus, common PEG settings are different from those associated with our work.

The third area of work we compare with is Stackelberg Security Games (SSG) [11], which model the interaction between defender and attacker as a game, then recommend patrol strategies for defenders against attackers. SSG has been successfully applied in security domains to generate randomized patrol strategies, e.g., to protect flights [11], for counter-terrorism and fare evasion checks on trains [12]. While the early work on SSG assumed a perfectly rational attacker, recent work has focused on attackers with bounded rationality and learning the parameters of the bounded rationality model using machine learning methods, such as maximum-likelihood estimation. An example of this approach is the PAWSmodel [13]. PAWS addresses the problem of learning poacher behavior within a game-theoretic interaction between defenders and poachers. Recent research has also made progress in designing patrol strategies against adversaries in graph settings [14]. In [15], patrol strategies against various types of adversaries are designed.

However, including various extensions, security games include an explicit model of the adversary, such as bounded rationality models and limited observation models. In general, in security games, a lack of sufficient data makes learning models of defender adversary interactions challenging. Distinct from these approaches, we do not model the adversary's decision making explicitly; rather, we learn the adversary interaction with the defender using real-world data. In our case, these are how the adversary moves from one patrol area to another and his or her probability of committing a crime given some patrol officers' presence.

A fourth thread of recent research combines machine learning with game theory. In [16], the defender's optimal strategy is generated in an 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.

The last area of work we compare with is on modeling opportunistic criminals. In [1], burglars' movement is modeled as a random walk, and in [2], a more general model of opportunistic criminals was proposed with algorithms for the optimal strategy against such criminals. Again, these papers include explicit models of the criminals and lack real-world data to learn the interactions.

3. Motivating Example

3.1. Domain Description

The motivating example for this study is the problem of controlling crime on a university campus. Our case study is about the American University, USC. 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's DPS, we have access to the crime report as well as patrol schedule on campus for three years (2011 to 2013). USC is a large enough university that we can claim that our methods are applicable to other campuses similar in size, for example malls.

USC's campus map is divided into five patrol areas, which is shown in Figure 1. DPS patrols in three shifts per day. Crime data are exclusively 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 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 an opportunity arises and there is insufficient presence of DPS officers.

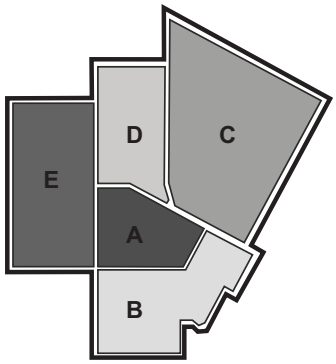


Figure 1. Campus map.

There are two reports that DPS shared with us. The first is about criminal activity that includes the 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 these data in Figure 2a. In this paper, 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 eight-hour patrol shift. Each crime data point contains five crime numbers, one for each patrol area.

The second dataset contains the DPS patrol allocation schedule. Every officer is allocated to patrolling within one patrol area. We show a snapshot of these data in Figure 2b. 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 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 are the number of crimes. Table 2 shows a sample of the summarized officer patrol allocation data, where rows correspond to shifts,

columns correspond to patrol areas and the numbers within a cell represent the number of patrol officers present. For example, from Table 2, we know that in Shift 1, the number of officers in area A is two, while the number of officers in areas B, C, D and E is one; while from Table 1, we know that in Shift 1, there was one crime each in areas A and B and two 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 areas the targets and each patrol shift a time step.

Table 1. Crime data for 3 shifts.

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 2. Patrol data for 3 shifts.

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

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

(**a**) 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

(**b**) Small patrol schedule for one shift

Figure 2. Sample data for (a) crime and (b) patrol respectively.

3.2. 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 be used to design optimal defender patrol allocations in any comparable urban crime setting. In the first cut, we model the learning problem as a Markov chain. Next, we use a DBN to model criminals' behavior and also present a compact form of DBN that leads to improved prediction. The DBN approach yields better results than the Markov chain approach. Finally, we present methods to find the optimal defender plan for the learned model with the frequent update of the criminal model.

4. Learning Model

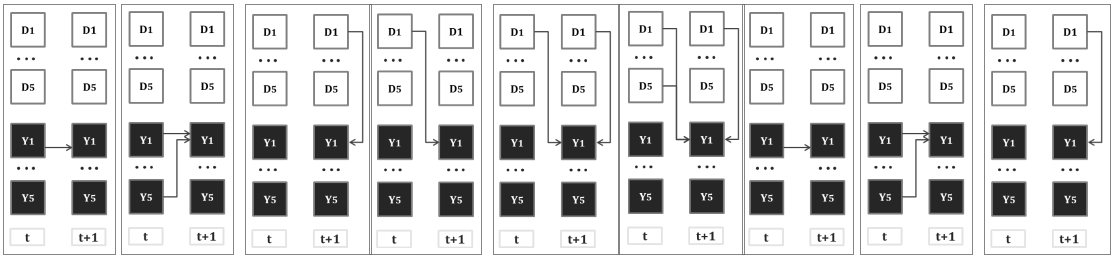
As stated earlier, we propose two approaches to learn the interaction between criminals and defenders: MCM and DBNM, which we explain in detail in the next two sub-sections. Both approaches are Markov process, but the first approach MCM learns the crime distribution using only observed data, while DBNM uses the number of criminals as an unobserved (hidden or latent) state. As a consequence, the learning algorithm for MCM uses simple maximum likelihood estimation, while DBNM uses the expectation maximization.

4.1. Markov Chain 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 to 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.

4.1.1. Crime Predicts Crime

In the first model shown in Figure 3a, we investigate the prediction of crime based on the crime distribution at the previous time step at the same target. This correlation is suggested based on the "crime predicts crime" ideas introduced in the criminology literature [17]. The desired correlation can be defined with the following mathematical function for all targets n: $Y_{n,t+1} = f(Y_{n,t})$.



(a) M1 **(b)** M2 **(c)** M3 **(d)** M4 **(e)** M5 **(f)** M6 **(g)** M7 **(h)** M8 **(i)** M9

Figure 3. Markov chain model (MCM) structures.

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 change 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, the probability for a sequence of events, i.e., Y_n, which refers to number of crimes over a sequence of time steps, can be calculated as follows:

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

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

$$l(A) = \log P(Y_n; A) = \sum_{i=1}^{|S_Y|} \sum_{j=1}^{|S_Y|} \sum_{t=1}^{T} 1\{Y_{n,t} = S_i \wedge Y_{n,t-1} = S_j\} \log A_{ij} \quad (2)$$

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

$$\max_A l(A) \text{ subject to } \sum_{i=1}^{|S_Y|} A_{i,j} = 1 \text{ for } j = 1...|S_Y|, \; A_{ij} \ge 0 \text{ for } i, j = 1...|S_Y| \quad (3)$$

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

$$\hat{A_{ij}} = \frac{\sum_{t=1}^{T} 1\{Y_t = S_i \wedge Y_{t-1} = S_j\}}{\sum_{t=1}^{T} 1\{Y_{t-1} = S_j\}} \quad (4)$$

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 Figure 3b includes the number of crimes at all other targets. This approach can be described with the following mathematical function: $Y_{n,t+1} = f(Y_{1:n,t})$.

4.1.2. 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 Figure 3c, $Y_{n,t+1} = f(D_{n,t+1})$, which means the effect of the defender at the same target and time step is considered; in Figure 3d, $Y_{n,t+1} = f(D_{n,t})$, which means the defender allocation at the previous step is considered; in Figure 3e, $Y_{n,t+1} = f(D_{n,t}, D_{n,t+1})$, which means the defender allocation in both the current and previous time step is considered; in Figure 3f, $Y_{n,t+1} = f(D_{1:n,t}, D_{n,t+1})$, 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.

4.1.3. 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 Figure 3g, $Y_{n,t+1} = f(D_{n,t+1}, Y_{n,t})$, which means that the distribution of the crime at the previous step and the defender allocation at the current step is considered; in Figure 3h, $Y_{n,t+1} = f(D_{n,t}, Y_{n,t})$; this has a similar structure as the previous one, except that it considers the defender allocation at the previous time step; in Figure 3i, $Y_{n,t+1} = f(D_{n,t}, D_{n,t+1}, Y_{n,t})$, that is the crime at that specific target is considered in addition to the defender allocation at the previous and current step.

4.2. 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 a 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.

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 Figure 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.

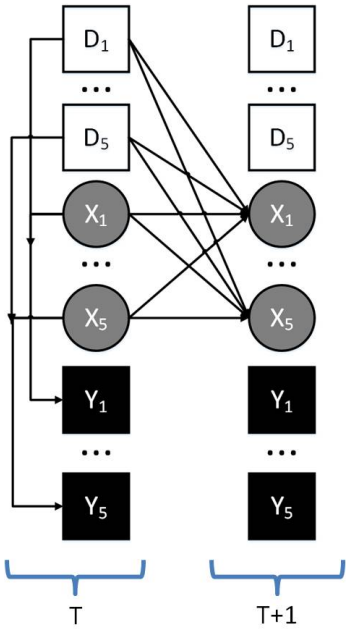


Figure 4. The Dynamic Bayesian Network (DBN) for games.

4.2.1. DBN Parameters

First, we introduce parameters that measure the 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: the 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: $\gamma(k,t) = P(x_t = k|y_1^T, d_1^T)$.
- Two-step prob.: $\xi(k,l,t) = P(x_t = k, x_{t+1} = l|y_1^T, d_1^T)$.

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.

4.2.2. Expectation Maximization

EM is a class of algorithms for finding maximum likelihood estimation for unknown parameters in DBN [18]. The EM algorithm has an initialization step, the Expectation (E) step and the 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 the E step. By iteratively performing the E and M steps, 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 [19].

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 following condition:

$$\sum_i \hat{\pi}(i) = 1, \sum_{x_{t+1}} \hat{A}(d_t, x_t, x_{t+1}) = 1, \sum_{y_t} \hat{B}(d_t, x_t, y_t) = 1 \tag{5}$$

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

Expectation Step: In the expectation step, we calculate 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 Equation (6). Next, we iteratively compute forward probability from Step 2 to T, which is shown in Equation (7). A similar technique works for backward probability: first, we calculate backward probability at Step T, $\beta(k,T)$, which is shown in Equation (8). Then, we recursively calculate backward probability from Step $T-1$ to 1, which is in Equation (9).

$$\alpha(k,1) = P(y_1, x_1 = k|d_1) = \hat{B}(d_1, k, Y_1) \cdot \hat{\pi}(k) \tag{6}$$

$$\begin{aligned}\alpha(k,t) &= P(y_1, y_2, ..., y_t, x_t = k|d_1, d_2, ..., d_t) \\ &= \sum_{x_{t-1}} \alpha(x_{t-1}, t-1)\hat{A}(d_{t-1}, x_{t-1}, x_t)\hat{B}(d_t, k, y_t)\end{aligned} \tag{7}$$

$$\beta(k,T) = 1 \tag{8}$$

$$\begin{aligned}&\beta(k,t) = P(y_{t+1}, y_{t+2}, ..., y_T|x_t = k, d_t, d_{t+1}, ..., d_T) = \\ &\sum_{x_{t+1}} \beta(x_{t+1}, t+1)\hat{B}(d_{t+1}, x_{t+1}, y_{t+1})\hat{A}(d_t, k, x_{t+1})\end{aligned} \tag{9}$$

Given the forward probability and the backward probability at each step, the total probability γ is computed as shown in Equation (10), and the two-step probability ξ is computed as shown in Equation (11).

$$\gamma(k,t) = P(x_t = k|y,d) = \frac{\alpha(k,t) \cdot \beta(k,t)}{\sum_k \alpha(k,t) \cdot \beta(k,t)} \tag{10}$$

$$\begin{aligned}\xi(k,l,t) &= P(x_t = k, x_{t+1} = l|y,d) \\ &= \frac{\alpha(k,t) \cdot A(d_t,k,l) \cdot \beta(l,t+1) \cdot B(d_{t+1},l,y_{t+1})}{\sum_k \alpha(k,t) \cdot \beta(k,t)}\end{aligned} \tag{11}$$

Maximization Step: In the maximization step, the estimate of π, A and B is updated using the probabilities we derive in the expectation step, as follows:

$$\hat{\pi}(k) = \gamma(k, 1) \tag{12}$$

$$\hat{A}(d, k, l) = \frac{\sum_t 1_{d_t=d} \xi(k, l, t)}{\sum_t \sum_l 1_{d_t=d} \xi(k, l, t)} \tag{13}$$

$$\hat{B}(d, x, y) = \frac{\sum_t 1_{y_t=y, d_t=d} \gamma(x, t)}{\sum_t 1_{d_t=d} \gamma(x, t)} \tag{14}$$

where $1_{d_t=d}$ is an indicator function: $1_{d_t=d} = 1$ when $d_t = d$ and zero 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, the criminal is x to the total number of situations where the defender is d and the criminal is x. To find a local optimal solution, the E and M steps are repeated, until $\hat{\pi}, \hat{A}, \hat{B}$ do not change significantly any more.

In the 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 depend 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 one (meaning $\leq$1) or two (meaning $\geq$2); the number of criminals/crimes is zero (no criminal/crime) or one ($\geq$1 criminal/crime). However, the number of unknown parameters is still exponential in N. As a concrete example, at USC, $N = 5$, and the number of unknown parameters are more than 32,768, 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.

4.2.3. EM on the Compact Model

In this part, we 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 [20] and our experimental results support our claims.

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 [6] (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 at USC and prior literature on criminology [1] 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 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 detail, 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 an N by one random vector indicating the number of crimes $Y_{i,t}$ at each target i at step t. D_t is an N by one random vector indicating the number of defenders $D_{i,t}$ at each target i at step t. X_t is an N by oen random vector indicating the number of criminals $X_{i,t}$ at each target i at step t.

Factored crime matrix: The number of crimes 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 $C^N \times C^N \times C^N$, we have a factored crime matrix of size $N \times C \times C \times C$. The first dimension of the 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 the marginal probability of the number of criminals at each target.

In the unmodified DBN, the distribution over all of the states at step t, $P(x_t)$ is a C^N by one vector. Additionally, the size of movement matrix A, which is the transition matrix from all of the input and hidden state combinations at the 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 are $P(X_i = k, t)$, which 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 the full state by multiplying the marginals at this step. Then, we get the marginals at the next step by evolving the recovered joint distribution of the state at the 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 the marginalized hidden state, we still need to recover the distribution of the 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 the properties of opportunistic criminals. Based on the crime reports and our discussion with DPS at 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 the current step, the criminals at each target independently decide the next target to go to, based on their target-specific observation of the 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 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 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 zero at location i at step $t+1$ is given by $\prod_{j=1}^{N} P(X_{t+1}^{j \to i} = 0)$.*

Proof 1. 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 extend Lemma 1 to any number of criminals at the next step using the concept of permutations; e.g., if there is one criminal at the next step, this means that there is only one station m where $X_{t+1}^{m \to i} = 1$, and for all other stations, $n \neq m$, $X_{t+1}^{n \to i} = 0$. One simple example is when $X_{i,t} \in \{0, 1, 2\}$; we can apply Lemma 1 to derive:

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

4.2.4. EMC^2 Procedure

The EM on CompaCtmodel (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.: $\gamma(i, k, t) = P(Y, X_{i,t} = k | D_1^T)$.
- Two-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 the 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:

$$\begin{aligned} & P(Y_1^{t-1}, X_{i,t} = 0 | D_1^t) \\ & = \sum_{X_{t-1}} P(Y_1^{t-1}, X_{i,t} = 0, X_{t-1} | D_1^t) \\ & = \sum_{X_{t-1}} P(Y_1^{t-1} | D_1^t, X_{i,t} = 0, X_{t-1}) P(X_{i,t} = 0 | D_1^t, X_{t-1}) P(X_{t-1} | D_1^t) \end{aligned} \quad (15)$$

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

$$\sum_{X_{t-1}} P(Y_1^{t-1} | D_1^t, X_{t-1}) P(X_{i,t} = 0 | D_{t-1}, X_{t-1}) P(X_{t-1} | D_1^t) \quad (16)$$

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

$$\sum_{X_{t-1}} P(Y_1^{t-1}, X_{t-1} | D_1^t) P(X_{i,t} = 0 | D_{t-1}, X_{t-1}) \quad (17)$$

Using the Boyen–Koller assumption in our compact model, we get:

$$P(Y_1^{t-1}, X_{t-1}|D_1^t) = \prod_j P(Y_1^{t-1}, X_{j,t-1}|D_1^t) \tag{18}$$

Furthermore, using Lemma 1, we get:

$$P(X_{i,t} = 0|D_{t-1}, X_{t-1}) = \prod_j P(X_t^{j\to i} = 0) \tag{19}$$

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

$$\sum_{X_{t-1}} \prod_j P(Y_1^{t-1}, X_{j,t-1}|D_1^t) P(X_t^{j\to i} = 0) \tag{20}$$

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 the principles of the generalized distributive law [21] 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 the distributive law, we reduce the computation for $P(Y_1^{t-1}, X_{i,t} = 0|D_1^t)$ by switching the sum and product:

$$\prod_j \sum_{X_{j,t-1}} P(Y_1^{t-1}, X_{j,t-1}|D_1^t) P(X_t^{j\to i} = 0) \tag{21}$$

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 Equations (7) to (9). In the following part, we provide the detail of EMC^2 procedure.

Initialization step: Set random initial conditions for criminal distribution π, transition matrix A and output matrix B. See Appendix A for an example of one such initialization.

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. This is explained below.

Step 1: Iteratively calculating forward probability

First, we calculate the forward probability at Step 1 ($\alpha(i, k, 1)$) as:

$$\begin{aligned}\alpha(i,k,1) &= P(Y_1, X_{i,1} = k|D_1) = P(Y_{i,1}, X_{i,1} = k|D_{i,1}) \\ &= P(Y_{i,1}|X_{i,1} = k, D_{i,1}) \cdot P(X_{i,1} = k|D_{i,1}) = B(i, D_{i,1}, k, Y_{i,1}) \cdot \pi(k, i)\end{aligned}$$

Then, we iteratively calculate the forward probability from Step 2 to T denoted as $\alpha(i, k, t)$. We then apply dynamic programming to go one step forward. In order to do this, we use $P(Y_1, Y_2, ..., Y_{t-1}, X_{i,t}|D_1, D_2, ..., D_{t-1})$ as an intermediate variable. Details are in the Appendix A.

Step 2: Iteratively calculating backward probability

First, we calculate backward probability at step T($\beta(i, k, T)$) as follows:

$$\beta(i, k, T) = 1$$

Then, we iteratively calculate the backward probability at each time step t ($\beta(i, k, t)$) from Step T-1 to 1. Detailed calculations are provided in Appendix A.

Step 3: Calculating total probability γ

The total probability $\gamma(i, k, t)$ is then calculated. Detailed calculations are shown in the Appendix.

Step 4: Calculating two-step probability ξ

Finally, we calculate the two-step probability ξ. We show the detailed calculations in the Appendix.

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:

$$\pi(k,i) = \gamma(i,k,1) \tag{22}$$

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:

$$B(i,d_{i,t},x_{i,t},y_{i,t}) = \frac{\sum_t 1_{Y_{i,t}=y_{i,t},D_{i,t}=d_{i,t}}\gamma(i,x_{i,t},t)}{\sum_t 1_{D_{i,t}=d_{i,t}}\gamma(i,x_{i,t},t)} \tag{23}$$

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

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

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 has been equal to y while the defender is d and the criminal is k over the expected total number of times at target i.

Step 3: Update transition matrix A:

$$A(m,d_{m,t},x_{m,t},i,x_{i,t+1}) = \frac{\sum_t 1_{D_t=d_t}\xi(m,d_{m,t},x_{m,t},i,x_{i,t+1}))}{\sum_t\sum_{x_{i,t+1}} 1_{D_t=d_t}\xi(m,d_{m,t},x_{m,t},i,x_{i,t+1}))} \tag{24}$$

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.

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

Computational complexity analysis: *EM on basic model:* In the basic model, the time complexity of the E step is $O(C^{2N}T)$. The time complexity for the M step is also $O(C^{2N}T)$. Thus, the computational complexity for the EM algorithm is $O(C^{2N}T)$; *EMC*^{2} *procedure:* In the EMC^2 procedure, the time complexity for the E step is $O(N^{C+1}T)$. α and β in Equations 11 and 13 are $O(N^C)$. Since we have to compute $\alpha(i,k,t)$ and $\beta(i,k,t)$ for all $i \in N$, $k = 1,2$ and $t \in T$, the computational complexity for forward and backward probability is $O(N^{C+1}T)$. For Equation (14), the complexity is $O(1)$, and the complexity for γ is $O(NT)$. For Equation (15), the complexity is $O(N)$, and the complexity for ξ is $O(N^{C+1}T)$. The total complexity for the E step is $O(N^{C+1}T + NT + N^{C+1}T) = O(N^{C+1}T)$. The time complexity for the M step is $O((C \cdot N)^2 T)$. Thus, the computational complexity for the EM algorithm is $O(N^{C+1}T + (C \cdot N)^2 T)$. Therefore, the EMC^2 procedure runs much faster than the EM in the basic model when C is small.

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 the criminals' behavior model and plan allocation strategies. Next, we present a slower optimal planning algorithm and a 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 themselves change in response to the deployed strategy.

To address the problem above, we propose an online planning mechanism. In this mechanism, we update the 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 1 presents the details of this mechanism.

Algorithm 1 Online planning ($Train\_data, T_u, T$).

$A, B, \pi \leftarrow Learn(Train\_data)$
$t = 0$
while $t < T$ **do**
  $[D_1, ..., D_{T_u}] \leftarrow Plan(A, B, \pi)$
  $[Y_1, ..., Y_{T_u}] \leftarrow Execute\{D_1, ..., D_{T_u}\}$
  $Train\_data \leftarrow Train\_data \cup \{D_1, Y_1, ..., D_{T_u}, Y_{T_u}\}$
  $A, B, \pi \leftarrow Update(Train\_data, A, B, \pi)$
  $t = t + T_u$
end while

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

5.1. Planning Algorithms

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

5.1.2. Brute Force Search

A naive way to solve this problem is to try all possible allocation strategies and to pick the one that leads to the 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.

5.1.3. 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.

Algorithm 2 DOGS (A, B, π).

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

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:

$$P_{j,1} = f_Y(\pi, D_1^j, B) P_{j,t} = \min_i [f_Y(f_X(A, X_{i,t-1}, D_{t-1}^i), D_t^j, B) + P_{i,t-1}] \tag{25}$$

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

$$\mathsf{Pa}[j,t] = \underset{i}{\text{argmin}} [f_Y(f_X(A, X_{i,t-1}, D_{t-1}^i), D_t^j, B) + P_{i,t-1}] \tag{26}$$

As P_{j,T_u} is the total number of crimes 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 $D_{T_u} = \text{argmin}_j P_{j,T_u}$. 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 the DOGS algorithm (Algorithm 2) is $O(C^{2N} T_u)$.

5.1.4. 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 the 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 for the strategy that optimizes the defender's utility at the current step (Algorithm 3). It finds the optimal patrol allocation D_t at the current step by minimizing the expected number of crimes 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 the T_u step. The complexity of greedy search is $O(C^N T_u)$.

Algorithm 3 Greedy $(A, B, \pi = X_1)$.

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 for**
4: **return** $D = [D_1, ..., D_{T_u}]$

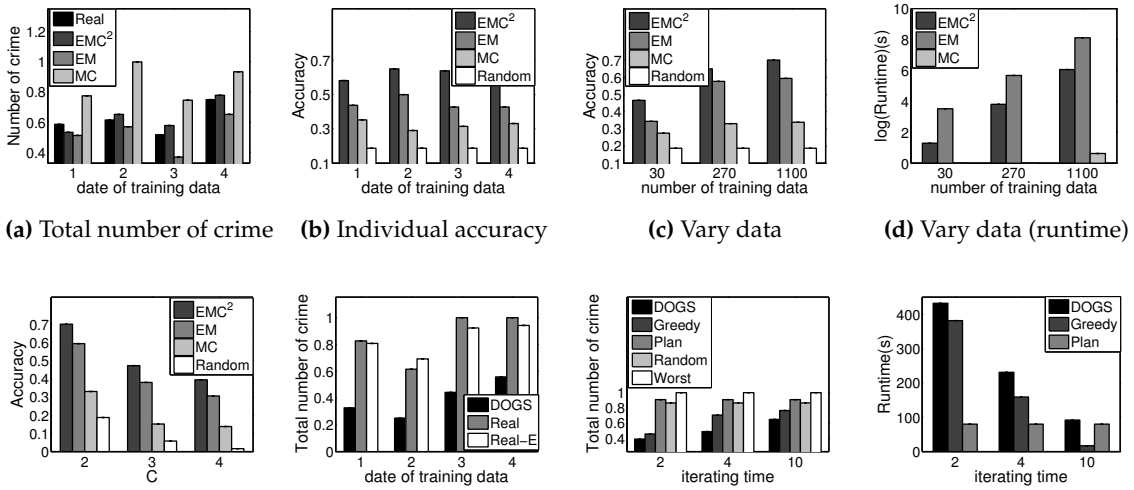
6. Experimental Results

6.1. Experimental Setup

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

6.2. Learning (Setting)

Our first experiment is on evaluating the performance of the EMC^2 algorithm in learning criminals' behavior. We use the case study of USC in our experiments. We obtained three years of crime reports and the corresponding patrol schedule followed at USC. Since the EMC^2 algorithm and the EM algorithm only reach the 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., the values of A, B and π, are generated by sampling values from a uniform random distribution over $[0, 1]$ for all of the elements and then normalizing the probabilities so that they satisfy the initial conditions. C is set to two by default, while the effect of varying C is compared in Figure 5e.



(a) Total number of crime **(b)** Individual accuracy **(c)** Vary data **(d)** Vary data (runtime)

(e) Vary C **(f)** Compare with real **(g)** Vary T_u **(h)** Vary T_u (Runtime)

Figure 5. Experimental results.

The results shown in Figure 5a compare the estimated numbers of crimes using different learning algorithms with the real number of crimes in one month. 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's data. That is why the estimated number of crimes for one month as the output by different learning algorithms is compared to the real number of crimes in that month. The x-axis in this figure indicates the index of the part of the data that we evaluate. The y-axis is the total number of crimes in the ninth month. The closer this number is to the real number of crimes, the better the prediction is. Three different algorithms are compared: (1) the Markov Chain (MC) algorithm, in which the best performance among all eight models is shown; (2) the exact EM algorithm; and (3) the EMC^2 algorithm. As can be seen, the prediction of EMC^2 is much closer compared to those of the EM and MC algorithms in all of the training groups. This indicates that the crime distribution is related to the criminals' location, and including the number of criminals at each target as a hidden state helps to improve the performance. In addition, the EMC^2 algorithm achieves better performance than EM by reducing the number of unknown variables to avoid over-fitting.

For Figure 5b, we measure the 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 5b, the y-axis represents the accuracy. The higher accuracy is, the more accurate our prediction is. We compare four different algorithms: the MC, EM and EMC^2 algorithms 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 that accuracy of the algorithms is data-independent.

We present additional results under this setting in Figure 5c,d. We compare the four approaches for varying the 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 are 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 5c, the 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 5d, the y-axis shows the runtime in seconds on a log scale. The more data we have, the longer it takes for each training method. The random algorithm is pre-generated and takes almost no time; hence, those data are not shown in the figure; the runtime for MC is negligible because the number of states is small ($O(4^N)$), and we traverse all of the data points only once; the runtime for the EMC^2 algorithm is significantly better than that for the EM algorithm, as is expected by our complexity analysis in Section 5.

In Figure 5e, 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 are 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 the number of crimes. Thus, the possibility of predicting an accurate number decreases. However, when C increases from three to four, the decrease in accuracy is small in EMC^2 due to the fact that data with a value of four rarely appear in both the crime and patrol dataset. This indicates that a small C is a good approximation. In addition, the EMC^2 algorithm again outperforms all other algorithms for any C.

In Figure 6, all of 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 in Figure 3. The y-axis represents the accuracy. We use four sets of data that are the same as Figure 5b to test the performance. All of the Markov chain models give similar to accuracy, which outperforms RP

significantly. However, as shown in Figure 5b, all of these models are outperformed by the EM and EMC^2 algorithms.

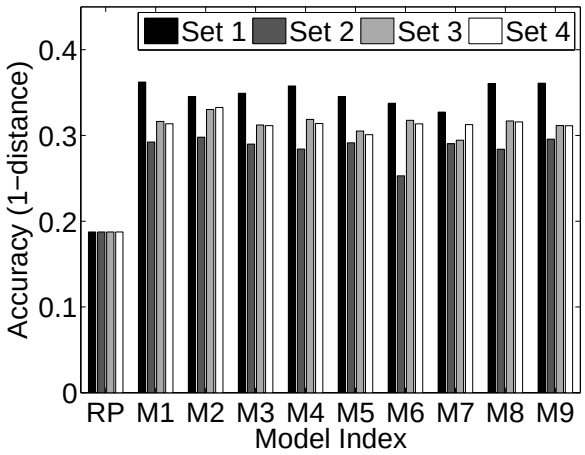


Figure 6. Markov chain models.

6.3. Learning and Planning (Real-World Data)

Figure 5f compares DOGS to the actual deployed allocation strategy generated by DPS experts at USC. Similar to the settings in Figure 5a, we divide the three-year data into four equal parts of nine months. For each part, we train on the first eight months's data using the EMC^2 algorithm and test different allocation strategies on the first 10 days of the ninth month's data. When testing the strategy, we assume the criminals' behavior remains unchanged during these 10 days. Three different scenarios are compared: (1) the real number of crimes, shown as real in Figure 5f; (2) the expected number of crimes with the 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 Figure 5f, the expected number of crimes with the DPS strategy is close to the real number of crimes, which indicates that EMC^2 captures the main features of the criminal behavior and provides a close estimate of the number of crimes. In addition, the DOGS algorithm outperforms the strategy generated by domain experts significantly. This demonstrates the effectiveness of the DOGS algorithm as compared to the current patrol strategy. By using the allocation strategy generated by DOGS, the total crime number reduces by $\sim$50% as compared to the currently-deployed strategy.

6.4. 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 [2], 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, which they use to learn an initial criminal behavior model. The criminal responds to the defenders' patrol schedule as predicted by the behavior model in [2]. Since the criminal behavior in [2] 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 dataset numbers (eight officers for five 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 the EMC^2 algorithm as the learning algorithm.

6.5. Learning and Planning Results

Figure 5g to Figure 7 present the results from our experiments about the online learning and planning mechanism. The four planning mechanisms that we consider are as follows: first, a random planning mechanism that randomly generates the 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 the DOGS algorithm; third, an online planning mechanism with a greedy planning algorithm that updates every T_u time steps; and the last mechanism is the online planning mechanism with the DOGS algorithm that also updates every T_u time steps.

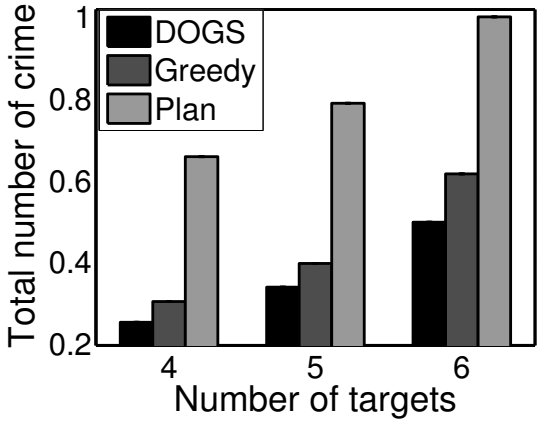


Figure 7. Varying N.

In Figure 5g, 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 protects the least valuable targets. The x-axis shows the update interval T_u, which is the time interval after which we update the criminals' behavior model. The y-axis is the expected number of crimes that happen under the deployed allocation strategy within 600 steps. The expected number of crimes under the pure planning mechanism stays the same with different T_u, because it does not update the criminals' model at all. For the 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, the DOGS algorithm outperforms the greedy algorithm. In Figure 5h, 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 the 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 7 shows the performance of the four planning mechanisms, but with different numbers 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 Figure 5g.

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

7. Real World Implementation

In this section, we introduce web-based software with two contributions. First, our system collects and analyzes crime reports and resources (security camera, emergency supplies, etc.) data, presenting them in various forms. Second, our patrol scheduler incorporates the learning and planning

algorithm we proposed before in a scheduling recommendation system. The software is currently under deployment and testing at the USC campus in Los Angeles, USA.

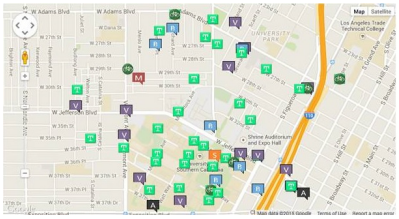
7.1. Multi-User Software

Our multi-user web-based software is built for the Department of Public Safety at the University of Southern California. It is composed of two main components: a data collector and a patrol scheduler. A detailed demonstration of our software can be found here (https://dl.dropboxusercontent.com/u/40377044/aamas_demo.pptx).

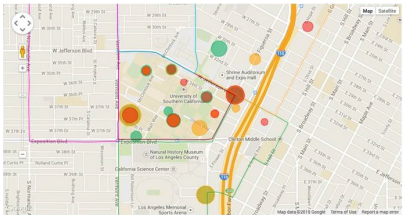
7.1.1. Data Collector

The data collector receives crime data from the police department and presents and analyzes them in various fashions. There are three main tasks of the data collector: First, it visualizes crime data with spatial and temporal information, as shown in Figure 8a, to help officers analyze the trend of crimes around campus. As shown in Figure 8b, 'hot' areas indicate attractive targets for criminals to commit crimes. As of now, the police department can cool down these hot spots by increasing patrol coverage when assigning officers in the field. However, with our planning approach, the police allocation will be more intelligent. As we also learn how criminals move in response to patrols, our approach does not simply cover the hot spots, but also allocates officers to areas where criminals are expected to move.

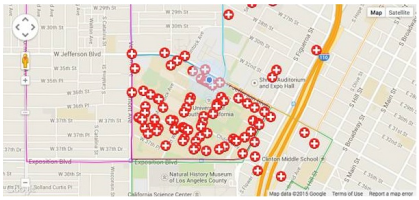
Second, the data collector provides information to the officers in the field about various available resources, such as emergency supplies and security cameras. As shown in Figure 8c, our software indicates the location for all of the emergency supplies on campus. Figure 8d shows the (mock) location of security cameras. To check certain locations, officers can use our software to watch the video from any camera. Finally, the data collector provides input for the patrol scheduler. By reading the data from the collector, the patrol scheduler can continuously learn and update criminals' behavior.



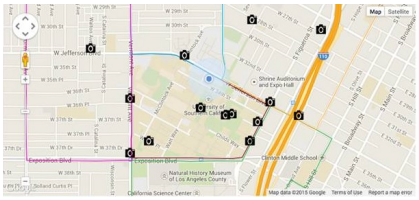
(a) Crime analysis



(b) Hot spot analysis



(c) Emergency supply



(d) Security cameras

Figure 8. Data collector.

7.1.2. Patrol Scheduler

Patrol settings: At USC, our approach further divides the enforcement area (encompassing the campus) into 18 patrol areas, which are shown in Figure 9. DPS patrols would be in shifts of 4 h each. At the beginning of each patrol shift, our algorithm assigns each available patrol officer to a patrol area, and the officer patrols this area in this shift.

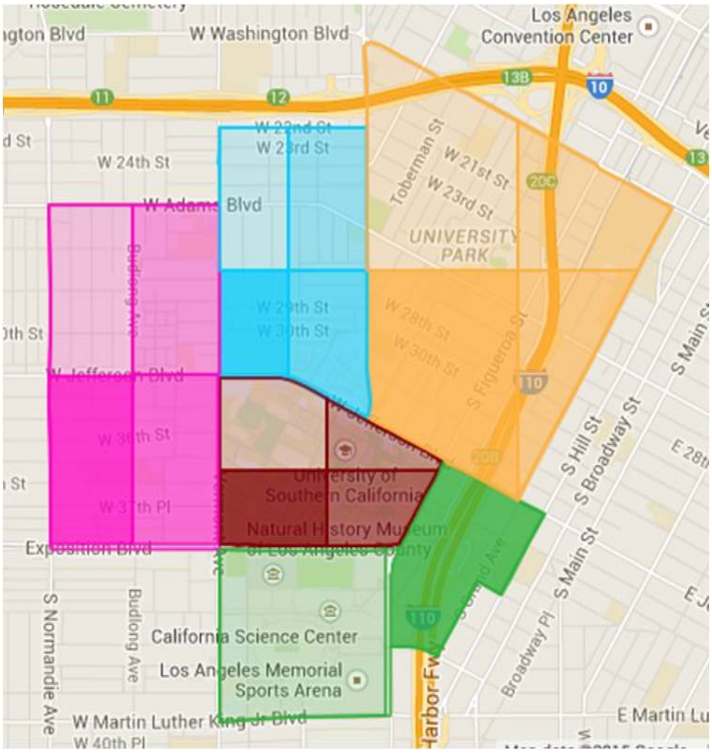


Figure 9. Patrol area.

Schedule generator: As part of the integration of our learning and planning with the software, we show the use of the EMC^2 algorithm to learn the criminals' behavior from previous crime and patrol data and the DOGS algorithm to generate patrol schedules for DPS officers. We highlight the recommended patrol area on the campus map, as shown in Figure 10.

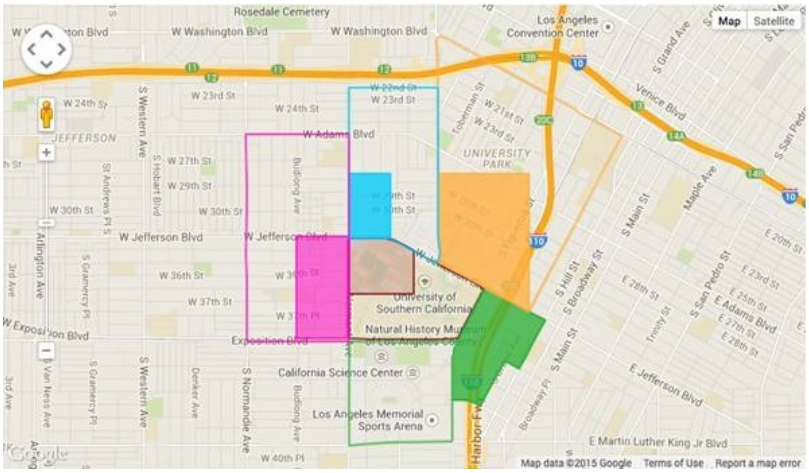


Figure 10. Sample patrol schedule.

8. Conclusions and Future Work

This paper introduces a novel framework to design patrol allocation against adaptive opportunistic criminals. First, we use Markov chain and DBN to model the interaction between officers and adaptive opportunistic criminals. 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 algorithms to keep pace with adaptive opportunistic criminals. Experimental validation with real-world data

supports our choice of model and assumptions. Further, our modeling assumptions were informed by inputs from our collaborators in the DPS at USC. We project a $\sim$50% reduction in the crime rate using our approach as opposed to the current approach followed by DPS.

In terms of future work, we would like to capture other patterns of crime using our Markov chain and DBN, such as how crime is influenced by the time of day, day of the week, season or domain factors, such whether or not USC classes are in session or whether it is football season, and other such factors. Another line of future work would be to do a sensitivity analysis of our results by considering various divisions of the three-year real-world data available at USC. Currently, we divide the dataset into four sets of nine months each. However, comparisons and analysis of the results based on three divisions of 12 months each or other divisions based on seasons and other factors would be an interesting line of future research.

Acknowledgments: This research is supported by MURI grant W911NF-11-1-0332. We thank Armorway. Inc. for helping generate the software.

Author Contributions: Chao Zhang, Shahrzad Gholami, Debarun Kar, Arunesh Sinha and Milind Tambe contributed to the design of the model, ran the experiment, interpreted results and wrote the manuscript. Manish Jain and Ripple Goyal contributed to generating the multi-user software.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A

Appendix A.1. EMC^2 Procedure Initialization Step

Example 1. A simple way to do the initialization is to use the uniform distribution, which is:

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

Appendix A.2. EMC^2 Procedure Expectation Step

Forward probability:

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

$$
\begin{aligned}
&P(Y_1, Y_2, ..., Y_{t-1}, X_{i,t}|D_1, D_2, ..., D_{t-1}) = \sum_{X_{t-1}} P(Y_1, Y_2, ..., Y_{t-1}, X_{t-1}, X_{i,t}|D_1, D_2, ..., D_{t-1}) \\
&= \sum_{X_{t-1}} P(Y_1, Y_2, ..., Y_{t-1}|X_{i,t}, X_{t-1}, D_1, D_2, ..., D_{t-1}) P(X_{i,t}|X_{t-1}, D_{t-1}) P(X_{t-1}) \\
&= \sum_{X_{t-1}} P(Y_1, Y_2, ..., Y_{t-1}|X_{t-1}, D_1, D_2, ..., D_{t-1}) P(X_{i,t}|X_{t-1}, D_{t-1}) P(X_{t-1}) \\
&= \sum_{X_{t-1}} P(Y_1, Y_2, ..., Y_{t-1}, X_{t-1}, D_1, D_2, ..., D_{t-1}) P(X_{i,t}|X_{t-1}, D_{t-1}) \\
&if\ X_{i,t} = 0 \\
&= \sum_{X_{t-1}} \prod_m \alpha(m, X_{m,t-1}, t-1) \prod_m P(X_{i,t} = 0|X_{m,t-1}, D_{m,t-1}) \\
&= \prod_m \sum_{X_{m,t-1}} \alpha(m, X_{m,t-1}, t-1) P(X_{i,t} = 0|X_{m,t-1}, D_{m,t-1}) \\
&if\ X_{i,t} = 1 \\
&= \sum_{X_{t-1}} \prod_m \alpha(m, X_{m,t-1}, t-1) [\prod_m \sum_{x_{i,t}} P(X_{i,t} = x_{i,t}|X_{m,t-1}, D_{m,t-1}) - \prod_m P(X_{i,t} = 0|X_{m,t-1}, D_{m,t-1})] \\
&= \prod_m \sum_{X_{m,t-1}} \alpha(m, X_{m,t-1}, t-1) \sum_{x_{i,t}} P(X_{i,t} = x_{i,t}|X_{m,t-1}, D_{m,t-1}) \\
&\quad - \prod_m \sum_{X_{m,t-1}} \alpha(m, X_{m,t-1}, t-1) P(X_{i,t} = 0|X_{m,t-1}, D_{m,t-1})
\end{aligned}
$$

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

$$
\begin{aligned}
&\alpha(i, k, t) = P(Y_1, Y_2, ..., Y_t, X_{i,t} = k|D_1, D_2, ..., D_t) \\
&= \sum_{X_t : X_{i,t} = k} P(Y_1, Y_2, ..., Y_t, X_t|D_1, D_2, ..., D_t) \\
&= \sum_{X_t : X_{i,t} = k} P(Y_1, Y_2, ..., Y_t|X_t, D_1, D_2, ..., D_t) P(X_t|D_1, D_2, ..., D_t) \\
&= \sum_{X_t : X_{i,t} = k} P(Y_1, Y_2, ..., Y_{t-1}|X_t, D_1, D_2, ..., D_t) P(Y_t|X_t, D_1, D_2, ..., D_t) P(X_t|D_1, D_2, ..., D_t) \\
&= \sum_{X_t : X_{i,t} = k} P(Y_1, Y_2, ..., Y_{t-1}|X_t, D_1, D_2, ..., D_t) P(Y_t|X_t, D_1, D_2, ..., D_t) P(X_t|D_1, D_2, ..., D_t) \\
&= \sum_{X_t : X_{i,t} = k} P(Y_1, Y_2, ..., Y_{t-1}, X_t|D_1, D_2, ..., D_{t-1}) P(Y_t|X_t, D_t) \\
&= \sum_{X_t : X_{i,t} = k} \prod_j P(Y_1, Y_2, ..., Y_{t-1}, X_{j,t}|D_1, D_2, ..., D_{t-1}) \prod_j P(Y_{j,t}|X_{j,t}, D_{j,t}) \\
&= \prod_{j \neq i} \sum_{X_{j,t}} P(Y_1, Y_2, ..., Y_{t-1}, X_{j,t}|D_1, D_2, ..., D_{t-1}) P(Y_{j,t}|X_{j,t}, D_{j,t}) \\
&\cdot P(Y_1, Y_2, ..., Y_{t-1}, X_{i,t} = k|D_1, D_2, ..., D_{t-1}) P(Y_{i,t}|X_{i,t} = k, D_{i,t})
\end{aligned}
$$

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

Backward probability:

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

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

Total probability:

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

In this step, we still use the transformation of the conditional probability.

Two-step probability:

$$\begin{aligned}
&\xi(m, x_{m,t}, i, x_{i,t+1}, t) = P(X_{m,t} = x_{m,t}, X_{i,t+1} = x_{i,t+1}|Y, D) \\
&= P(Y|x_{m,t}, x_{i,t+1}, D) \cdot P(x_{i,t+1}|x_{m,t}, D) \cdot P(x_{m,t}, D) \\
&= P(Y_1, .., Y_t|x_{m,t}, x_{i,t+1}, D) \cdot P(Y_{t+1}, ..., Y_T|x_{m,t}, x_{i,t+1}, D) \cdot P(x_{m,t}, D)P(x_{i,t+1}|x_{m,t}) \\
&= P(Y_1, .., Y_t, x_{m,t}, D) \cdot P(Y_{t+1}, ..., Y_T|x_{i,t+1}, D)P(x_{i,t+1}|x_{m,t}) \\
&= P(Y_1, .., Y_t, x_{m,t}, D) \cdot \sum_{X_{t+1}:X_{i,t+1}=x_{i,t+1}} P(Y_{t+2}, ..., Y_T|X_{t+1}, D)P(Y_{t+1}|X_{t+1}, D)P(x_{i,t+1}|x_{m,t}) \\
&= P(Y_1, .., Y_t, x_{m,t}, D) \cdot P(X_{i,t+1}|X_{m,t})P(Y_{t+2}, ..., Y_T|X_{i,t+1}, D)P(Y_{i,t+1}|X_{i,t+1}, D) \\
&\prod_{j\neq i} \sum_{X_{j,t+1}} P(Y_{t+2}, ..., Y_T|X_{j,t+1}, D)P(Y_{j,t+1}|X_{j,t+1}, D)P(X_{j,t+1}|X_{m,t}) \\
&= \alpha(m, x_{m,t}, t) \cdot A(i, D_{i,t}, X_{i,t}, m, X_{m,t+1})\beta(j, X_{i,t+1}, t+1) \\
&\prod_{j\neq i} \sum_{X_{j,t+1}} (\beta(j, X_{j,t+1}, t+1)B(j, D_{j,t+1}, X_{j,t+1}, Y_{j,t+1})A(i, D_{i,t}, X_{i,t}, j, X_{j,t+1})) \qquad \text{(A1)}
\end{aligned}$$

This calculation is similar to the way we calculate γ. From the second line to the 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 $\frac{1}{P(Y,D)}$.

References

1. Short, M.B.; D'orsogna, M.R.; Pasour, V.B.; Tita, G.E.; Brantingham, P.J.; Bertozzi, A.L.; Chayes, L.B. A statistical model of criminal behavior. *Math. Models Methods Appl. Sci.* **2008**, *18*, 1249–1267.
2. Zhang, C.; Jiang, A.X.; Short, M.B.; Brantingham, P.J.; Tambe, M. Defending against opportunistic criminals: New game-theoretic frameworks and algorithms. In *Decision and Game Theory for Security*; Springer: Los Angeles, CA, USA, 2014; pp. 3–22.
3. Jain, M.; Tsai, J.; Pita, J.; Kiekintveld, C.; Rathi, S.; Tambe, M.; Ordóñez, F. Software assistants for randomized patrol planning for the lax airport police and the federal air marshal service. *Interfaces* **2010**, *40*, 267–290.
4. Shieh, E.; An, B.; Yang, R.; Tambe, M.; Baldwin, C.; DiRenzo, J.; Maule, B.; Meyer, G. PROTECT: A deployed game theoretic system to protect the ports of the United States. In Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems, Valencia, Spain, 4–8 June 2012; International Foundation for Autonomous Agents and Multiagent Systems: Valencia, Spain, 2012; Volume 1; pp. 13–20.
5. Chen, H.; Chung, W.; Xu, J.J.; Wang, G.; Qin, Y.; Chau, M. Crime data mining: A general framework and some examples. *Computer* **2004**, *37*, 50–56.
6. Boyen, X.; Koller, D. Tractable inference for complex stochastic processes. In Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence, Madison, WI, USA, 24–26 July 1998; Morgan Kaufmann Publishers Inc.: San Francisco, CA, USA, 1998; pp. 33–42.
7. Nath, S.V. Crime pattern detection using data mining. In Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology Workshops, WI-IAT 2006 Workshops, Hong Kong, China, 18–22 December 2006; IEEE: Hong Kong, China, 2006; pp. 41–44.
8. De Bruin, J.S.; Cocx, T.K.; Kosters, W.A.; Laros, J.F.; Kok, J.N. Data mining approaches to criminal career analysis. Sixth International Conference on Data Mining, ICDM'06, Hong Kong, China, 18–22 December 2006; IEEE: Hong Kong, China, 2006; pp. 171–177.
9. Oatley, G.; Ewart, B.; Zeleznikow, J. Decision support systems for police: Lessons from the application of data mining techniques to soft forensic evidence. *Artif. Intell. Law* **2006**, *14*, 35–100.
10. Hespanha, J.P.; Prandini, M.; Sastry, S. Probabilistic pursuit-evasion games: A one-step nash approach. In Proceedings of the 39th IEEE Conference on Decision and Control, Sydney, NSW, Australia, 12–15 December 2000; IEEE: Sydney, NSW, Australia, 2000; Volume 3, pp. 2272–2277.
11. Tambe, M. *Security and Game Theory: Algorithms, Deployed Systems, Lessons Learned*; Cambridge University Press: Cambridge, UK, 2011.

12. Jiang, A.X.; Yin, Z.; Zhang, C.; Tambe, M.; Kraus, S. Game-theoretic randomization for security patrolling with dynamic execution uncertainty. In Proceedings of the 2013 International Conference on Autonomous Agents and Multi-Agent Systems, Saint Paul, MN, USA, 6–10 May 2013; International Foundation for Autonomous Agents and Multiagent Systems: Saint Paul, MN, USA, 2013; pp. 207–214.
13. Yang, R.; Ford, B.; Tambe, M.; Lemieux, A. Adaptive resource allocation for wildlife protection against illegal poachers. In Proceedings of the 2014 International Conference on Autonomous Agents and Multi-Agent Systems, Paris, France, 5–9 May 2014; International Foundation for Autonomous Agents and Multiagent Systems: Paris, France, 2014; pp. 453–460.
14. Basilico, N.; Gatti, N.; Amigoni, F. Leader-follower strategies for robotic patrolling in environments with arbitrary topologies. In Proceedings of the 8th International Conference on Autonomous Agents and Multiagent Systems, Budapest, Hungary, 10–15 May 2009; International Foundation for Autonomous Agents and Multiagent Systems: Budapest, Hungary, 2009; Volume 1, pp. 57–64.
15. Basilico, N.; Gatti, N.; Rossi, T.; Ceppi, S.; Amigoni, F. 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, Milan, Italy, 15–18 September 2009; IEEE Computer Society: Washington, DC, USA, 2009; Volume 2, pp. 557–564.
16. Blum, A.; Haghtalab, N.; Procaccia, A.D. Learning optimal commitment to overcome insecurity. In Proceedings of the 28th Annual Conference on Neural Information Processing Systems (NIPS), Montreal, QC, Canada, 8–13 December 2014.
17. Malik, A.; Maciejewski, R.; Towers, S.; McCullough, S.; Ebert, D.S. Proactive spatiotemporal resource allocation and predictive visual analytics for community policing and law enforcement. *Vis. Comput. Graph.* **2014**, *20*, 1863–1872.
18. Dempster, A.P.; Laird, N.M.; Rubin, D.B. Maximum likelihood from incomplete data via the EM algorithm. *J. R. Stat. Soc. Ser. B (Methodol.)* **1977**, *39*, 1–38.
19. Bishop, C.M. *Pattern Recognition and Machine Learning (Information Science and Statistics)*; Springer-Verlag New York, Inc.: Secaucus, NJ, USA, 2006.
20. Blumer, A.; Ehrenfeucht, A.; Haussler, D.; Warmuth, M.K. Occam's Razor. *Inf. Process. Lett.* **1987**, *24*, 377–380.
21. Aji, S.M.; McEliece, R.J. The generalized distributive law. *IEEE Trans. Inf. Theory* **2000**, *46*, 325–343.