Journal of Artificial Intelligence Research 48 (2013) 583-634 Submitted 4/13; published 11/13

Protecting Moving Targets with Multiple Mobile Resources

Fei Fang FEIFANG@USC.EDU
Albert Xin Jiang JIANGX@USC.EDU
Milind Tambe TAMBE@USC.EDU
University of Southern California
Los Angeles, CA 90089 USA

Abstract

In recent years, Stackelberg Security Games have been successfully applied to solve resource allocation and scheduling problems in several security domains. However, previous work has mostly assumed that the targets are stationary relative to the defender and the attacker, leading to discrete game models with finite numbers of pure strategies. This paper in contrast focuses on protecting mobile targets that leads to a continuous set of strategies for the players. The problem is motivated by several real-world domains including protecting ferries with escort boats and protecting refugee supply lines. Our contributions include: (i) A new game model for multiple mobile defender resources and moving targets with a discretized strategy space for the defender and a continuous strategy space for the attacker. (ii) An efficient linear-programming-based solution that uses a compact representation for the defender's mixed strategy, while accurately modeling the attacker's continuous strategy using a novel sub-interval analysis method. (iii) Discussion and analysis of multiple heuristic methods for equilibrium refinement to improve robustness of defender's mixed strategy. (iv) Discussion of approaches to sample actual defender schedules from the defender's mixed strategy. (iv) Detailed experimental analysis of our algorithms in the ferry protection domain.

1. Introduction

In the last few years, game-theoretic decision support systems have been successfully deployed in several domains to assist security agencies (defenders) in protecting critical infrastructure such as ports, airports and air-transportation infrastructure (Tambe, 2011; Gatti, 2008; Marecki, Tesauro, & Segal, 2012; Jakob, Vaněk, & Pěchouček, 2011). These decision support systems assist defenders in allocating and scheduling their limited resources to protect targets from adversaries. In particular, given limited security resources it is not possible to cover or secure all target at all times; and simultaneously, because the attacker can observe the defender's daily schedules, any deterministic schedule by the defender can be exploited by the attacker (Paruchuri, Tambe, Ordóñez, & Kraus, 2006; Kiekintveld, Islam, & Kreinovich, 2013; Vorobeychik & Singh, 2012; Conitzer & Sandholm, 2006).

One game-theoretic model that has been deployed to schedule security resources in such domains is that of a Stackelberg game between a leader (the defender) and a follower (the attacker). In this model, the leader commits to a *mixed strategy*, which is a randomized schedule specified by a probability distribution over deterministic schedules; the follower then observes the distribution and plays a best response (Korzhyk, Conitzer, & Parr, 2010). Decision-support systems based on this model have been successfully deployed, including ARMOR at the LAX airport (Pita, Jain, Marecki, Ordóñez, Portway, Tambe, Western, Paruchuri, & Kraus, 2008), IRIS for the US Federal Air Marshals service (Tsai, Rathi, Kiekintveld, Ordonez, & Tambe, 2009), and PROTECT for the US Coast Guard (Shieh, An, Yang, Tambe, Baldwin, DiRenzo, Maule, & Meyer, 2012).

Most previous work on game-theoretic models for security has assumed either stationary targets such as airport terminals (Pita et al., 2008), or targets that are stationary relative to the defender and the attacker, e.g., trains (Yin, Jiang, Johnson, Kiekintveld, Leyton-Brown, Sandholm, Tambe, & Sullivan, 2012) and planes (Tsai et al., 2009), where the players can only move along with the targets to protect or attack them). This stationary nature leads to discrete game models with finite numbers of pure strategies. In this paper we focus on security domains in which the defender needs to protect a mobile set of targets. The attacker can attack these targets at any point in time during their movement, leading to a continuous set of strategies. The defender can deploy a set of mobile escort resources (called patrollers for short) to protect these targets. We assume the game is zero-sum, and allow the values of the targets to vary depending on their locations and time. The defender's objective is to schedule the mobile escort resources to minimize attacker's expected utility. We call this problem Multiple mobile Resources protecting Moving Targets (MRMT).

The first contribution of this paper is a novel game model for MRMT called MRMT_{sg}. MRMT_{sg} is an attacker-defender Stackelberg game model with a continuous set of strategies for the attacker. In contrast, while the defender's strategy space is also continuous, we discretize it in MRMT_{sg} for three reasons. Firstly, if we let the defender's strategy space to be continuous, the space of mixed strategies for the defender would then have infinite dimensions, which makes exact computation infeasible. Secondly, in practice, the patrollers are not able to have such fine-grained control over their vehicles, which makes the actual defender's strategy space effectively a discrete one. Finally, the discretized defender strategy space is a subset of the original continuous defender strategy space, so the optimal solution calculated under our formulation is a feasible solution in the original game and gives a lower-bound guarantee for the defender in terms of expected utility for the original continuous game. On the other hand, discretizing the attacker's strategy space can be highly problematic as we will illustrate later in this paper. In particular, if we deploy a randomized schedule for the defender under the assumption that the attacker could only attack at certain discretized time points, the actual attacker could attack at some other time point, leading to a possibly worse outcome for the defender.

Our second contribution is CASS (Solver for Continuous Attacker Strategies), an efficient linear program to exactly solve MRMT_{sg}. Despite discretization, the defender strategy space still has an exponential number of pure strategies. We overcome this shortcoming by compactly representing the defender's mixed strategies as marginal probability variables. On the attacker side, CASS exactly and efficiently models the attacker's continuous strategy space using *sub-interval analysis*, which is based on the observation that given the defender's mixed strategy, the attacker's expected utility is a piecewise-linear function. Along the way to presenting CASS, we present DASS (Solver for Discretized Attacker Strategies), which finds minimax solutions for MRMT_{sg} games while constraining the attacker to attack at discretized time points. For clarity of exposition we first derive DASS and CASS for the case where the targets move on a one-dimensional line segment. We later show that DASS and CASS can be extended to the case where targets move in a two-dimensional space.

Our third contribution is focused on equilibrium refinement. Our game has multiple equilibria, and the defender strategy found by CASS can be suboptimal with respect to uncertainties in the attacker's model, e.g., if the attacker can only attack during certain time intervals. We present two heuristic equilibrium refinement approaches for this game. The first, *route-adjust*, iteratively computes a defender strategy that dominates earlier strategies. The second, *flow-adjust*, is a linear-programming-based approach. Our experiments show that flow-adjust is computationally faster than route-adjust but route-adjust is more effective in selecting robust equilibrium strategies.

Additionally, we provide several sampling methods for generating practical patrol routes given the defender strategy in compact representation. Finally we present detailed experimental analyses of our algorithm in the ferry protection domain. CASS has been deployed by the US Coast Guard since April 2013.

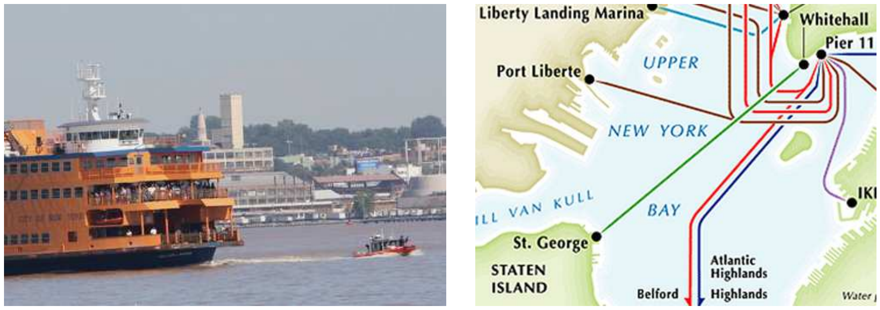
The rest of the article is organized as follows: Section 2 provides our problem statement. Section 3 presents the MRMT_{sg} model and an initial formulation of the DASS and CASS for a one-dimensional setting. Section 4 discusses equilibrium refinement, followed by Section 5 which gives the generalized formulation of DASS and CASS for two-dimensional settings. Section 6 describes how to sample a patrol route and Section 7 provides experimental results in the ferry protection domain. Section 8 discusses related work, followed by Section 9, which provides concluding remarks, and Section 10, which discusses future work. At the end of the article, Appendix A provides a table listing all the notations used in the article, and Appendix B provides the detailed calculation for finding the intersection points in the 2-D case.

2. Problem Statement

One major example of the practical domains motivating this paper is the problem of protecting ferries that carry passengers in many waterside cities. Packed with hundreds of passengers, these may present attractive targets for an attacker. For example, the attacker may ram a suicide boat packed with explosives into the ferry as happened with attacks on French supertanker Limburg and USS Cole (Greenberg, Chalk, & Willis, 2006). In this case, the intention of the attacker can only be detected once he gets very close to the ferry. Small, fast and well-armed patrol boats (patrollers) can provide protection to the ferries (Figure 1(a)), by detecting the attacker and stopping him with the armed weapons. However, there are often limited numbers of patrol boats, i.e., they cannot protect the ferries at all times at all locations. We first focus on the case where ferries and patrol boats move in a one-dimensional line segment (this is a realistic setting and also simplifies exposition); we will discuss the two-dimensional case in Section 5.

2.1 Domain Description

In this problem, there are L moving targets, $F_1, F_2, ..., F_L$. We assume that these targets move along a one-dimensional domain, specifically a straight line segment linking two terminal points which we will name A and B. This is sufficient to capture real-world domains such as ferries moving back-and-forth in a straight line between two terminals as they do in many ports around the world; an example is the green line shown in Figure 1(b). We will provide an illustration of our geometric formulation of the problem in Figure 2.1. The



(a) (b)

Figure 1: (a) Protecting ferries with patrol boats; (b) Part of the map of New York Harbor Commuter Ferry Routes. The straight line linking St. George Terminal and Whitehall Terminal indicates a public ferry route run by New York City Department of Transportation.

targets have fixed daily schedules. The schedule of each target can be described as a continuous function $S_q : T \to D$ where $q = 1, ..., L$ is the index of the target, $T = [0, 1]$ is a continuous time interval (e.g., representing the duration of a typical daily patrol shift) and $D = [0, 1]$ is the continuous space of possible locations (normalized) with 0 corresponding to terminal A and 1 to terminal B. Thus $S_q(t)$ denotes the position of the target F_q at a specified time t. We assume S_q is piecewise linear.

The defender has W mobile patrollers that can move along D to protect the targets, denoted as $P_1, P_2, ..., P_W$. Although capable of moving faster than the targets, they have a maximum speed of v_m. While the defender attempts to protect the targets, the attacker will choose a certain time and a certain target to attack. (In the rest of the paper, we denote the defender as "she" and the attacker as "he"). The probability of attack success depends on the positions of the patrollers at that time. Specifically, each patroller can detect and try to intercept anything within the *protection radius* r_e but cannot detect the attacker prior to that radius. Thus, a patroller protects all targets within her protective circle of radius r_e (centered at her current position), as shown in Figure 2.1.

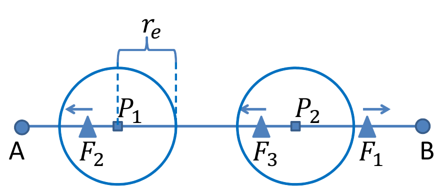


Figure 2: An example with three targets (triangles) and two patrollers (squares). The protective circles of the patrollers are shown with protection radius r_e. A patroller protects all targets in her protective circle. Patroller P_1 is protecting F_2 and P_2 is protecting F_3.

Symmetrically, a target is protected by all patrollers whose protective circles can cover it. If the attacker attacks a protected target, then the probability of successful attack is a decreasing function of the number of patrollers that are protecting the target. Formally, we use a set of coefficients $\{C_G\}$ to describe the strength of the protection.

Definition 1. *Let $G \in \{1, ..., W\}$ be the total number of patrollers protecting a target F_q, i.e., there are G patrollers such that F_q is within radius r_e of each of the G patrollers. Then $C_G \in [0, 1]$ specifies the probability that the patrollers can successfully stop the attacker. We require that $C_{G_1} \leq C_{G_2}$ if $G_1 \leq G_2$, i.e., more patrollers offer better protection.*

As with previous work in security games (Tambe, 2011; Yin et al., 2012; Kiekintveld, Jain, Tsai, Pita, Ordóñez, & Tambe, 2009), we model the game as a Stackelberg game, where the defender commits to a randomized strategy first, and then the attacker can respond to such a strategy. The patrol schedules in these domains were previously created by hand; and hence suffer the drawbacks of hand-drawn patrols, including lack of randomness (in particular, informed randomness) and reliance on simple patrol patterns (Tambe, 2011), which we remedy in this paper.

2.2 Defender Strategy

A pure strategy of the defender is to designate a movement schedule for each patroller. Analogous to the target's schedule, a patroller's schedule can be written as a continuous function $R_u : T \to D$ where $u = 1, ..., W$ is the index the patroller. R_u must be compatible with the patroller's velocity range. A mixed defender strategy is a randomization over the pure strategies, denoted as f.

2.3 Attacker Strategy

The attacker conducts surveillance of the defender's mixed strategy and the targets' schedules; he may then execute a pure strategy response to attack a certain target at a certain time. The attacker's pure strategy can be denoted as $\langle q, t \rangle$ where q is the index of target to attack and t is the time to attack.

2.4 Utility Function

We assume the game is zero-sum. If the attacker performs a successful attack on target F_q at location x at time t, he gets a positive reward $U_q(x, t)$ and the defender gets $-U_q(x, t)$, otherwise both players get utility zero. The positive reward $U_q(x, t)$ is a known function which accounts for many factors in practice. For example, an attacker may be more effective in his attack when the target is stationary (such as at a terminal point) than when the target is in motion. As the target's position is decided by the schedule, the utility function can be written as $U_q(t) \equiv U_q(S_q(t), t)$. We assume that for each target F_q, $U_q(t)$ can be represented as a piecewise linear function of t.

2.5 Equilibrium

Since our game is zero-sum, the Strong Stackelberg Equilibrium can be calculated by finding the minimax/maximin strategy (Fudenberg & Tirole, 1991; Korzhyk et al., 2010). That is, we can find the optimal defender strategy by finding a strategy that minimizes the maximum of attacker's expected utility.

Definition 2. *For single patroller case, the attacker expected utility of attacking target F_q at time t given defender mixed strategy f is*

$$AttEU_f(F_q, t) = (1 - C_1\omega_f(F_q, t))U_q(t) \tag{1}$$

$U_q(t)$ is the reward for a successful attack, $\omega_f(F_q, t)$ is the probability that the patroller is protecting target F_q at time t and C_1 is the protection coefficient of single patroller. We drop the subscript if f is obvious from the context. As C_1 and $U_q(t)$ are constants for a given attacker's pure strategy $\langle q, t\rangle$, $\mathrm{AttEU}(F_q, t)$ is purely decided by $\omega(F_q, t)$. The definition with multiple patrollers will be given in Section 3.4. We further denote the attacker's maximum expected utility as

$$\mathrm{AttEU}_f^m = \max_{q,t} \mathrm{AttEU}_f(F_q, t) \tag{2}$$

So the optimal defender strategy is a strategy f such that the AttEU_f^m is minimized, formally

$$f \in \arg\min_{f'} \mathrm{AttEU}_{f'}^m \tag{3}$$

2.6 Assumptions

In our problem, the following assumptions are made based on discussions with domain experts. Here we provide our justifications for these assumptions. While appropriate for the current domain of application, relaxing these assumptions for future applications remains an issue for future work; and we provide an initial discussion in Section 10.

- *The attacker's plan is decided off-line, i.e., the attacker does not take into account the patroller's current partial route (partial pure strategy) in executing an attack:* This assumption is similar to the assumption made in other applications of security games and justified elsewhere (An, Kempe, Kiekintveld, Shieh, Singh, Tambe, & Vorobeychik, 2012; Pita, Jain, Ordonez, Portway, Tambe, Western, Paruchuri, & Kraus, 2009; Tambe, 2011). One key consideration is that given that attackers have limited resources as well, for them to generate and execute complex conditional plans that change based on "on-line" observations of defender's pure strategy is both difficult and risky.

- *A single attacker is assumed instead of multiple attackers:* This assumption arises because performing even a single attack is already costly for the attacker. Thus, having coordinating attackers at the same time will be even harder and therefore significantly less likely for the attacker.

- *The game is assumed to be zero-sum:* In this case, the objectives of the defender and attacker are in direct conflict: preventing an attack with higher potential damage is a bigger success to the defender in our game.

- *The schedules for the targets are deterministic:* For the domains we focus on, potential delays in the targets' schedules are usually within several minutes if any, and the targets will try to catch up with the fixed schedules as soon as possible. Therefore, even when delays occur, the deterministic schedule for a target can be viewed as a good approximation of the actual schedule.

3. Models

In this section, we introduce our MRMT$_{\text{sg}}$ model that uses a discretized strategy space for the defender and a continuous strategy space for the attacker. For clarity of exposition, we then introduce the DASS approach to compute a minimax solution for discretized attacker strategy space (Section 3.2), followed by CASS for the attacker's continuous strategy space (Section 3.3). We first assume a single patroller in Sections 3.1 through 3.3 and then generalize to multiple patrollers in Section 3.4.

3.1 Representing Defender's Strategies

In this subsection, we introduce the discretized defender strategy space and the compact representation used to represent the defender's mixed strategy. We show that the compact representation is equivalent to the intuitive full representation, followed by several properties of the compact representation.

Since the defender's strategy space is discretized, we assume that each patroller only makes changes at a finite set of time points $T = \{t_1, t_2, ..., t_M\}$, evenly spaced across the original continuous time interval. $t_1 = 0$ is the starting time and $t_M = 1$ is the normalized ending time. We denote by δt the distance between two adjacent time points: $\delta t = t_{k+1} - t_k = \frac{1}{M-1}$. We set δt to be small enough such that for each target F_q, the schedule $S_q(t)$ and the utility function $U_q(t)$ are linear in each interval $[t_k, t_{k+1}]$ for $k = 1, \ldots, M-1$, i.e., the target is moving with uniform speed and the utility of a successful attack on it changes linearly during each of these intervals. Thus, if t_0 is a breakpoint of $S_q(t)$ or $U_q(t)$ for any q, it can be represented as $t_0 = \delta t K_0$ where K_0 is an integer.

In addition to discretization in time, we also discretize the line segment AB that the targets move along into a set of points $D = \{d_1, d_2, ..., d_N\}$ and restrict each patroller to be located at one of the discretized points d_i at any discretized time point t_k. Note that D is not necessarily evenly distributed and the targets' locations are not restricted at any t_k. During each time interval $[t_k, t_{k+1}]$, each patroller moves with constant speed from her location d_i at time t_k to her location d_j at time t_{k+1}. Only movements compatible with the speed limit v_m are possible. The points $d_1, d_2, ..., d_N$ are ordered by their distance to terminal A, and d_1 refers to A and d_N refers to B. Since the time interval is discretized into M points, a patroller's route R_u can be represented as a vector $R_u = (d_{r_u(1)}, d_{r_u(2)}, ..., d_{r_u(M)})$. $r_u(k)$ indicates the index of the discretized distance point where the patroller is located at time t_k.

As explained in Section 1, we discretized the defender's strategy space not only for computational reasons. It is not even clear whether an equilibrium exists in the original game with continuous strategy space for both players. The discretization is made also because of the practical constraint of patrollers.

For expository purpose, we first focus on the case with a single defender resource and then generalize to larger number of resources later. For a single defender resource, the defender's mixed strategy in *full representation* assigns a probability to each of the patrol routes that can be executed. Since at each time step a patroller can choose to go to at most N different locations, there are at most N^M possible patrol routes in total and this number is achievable if there is no speed limit (or v_m is large enough). The exponentially growing number of routes will make any analysis based on full representation intractable.

Therefore, we use the *compact representation* of the defender's mixed strategy.

Definition 3. *The* compact representation *for a single defender resource is a compact way to represent the defender's mixed strategy using* flow distribution variables $\{f(i,j,k)\}$. $f(i,j,k)$ *is the probability of the patroller moving from* d_i *at time* t_k *to* d_j *at time* t_{k+1}.

The complexity of the compact representation is $O(MN^2)$, which is much more efficient compared to the full representation.

Proposition 1. *Any strategy in full representation can be mapped into a compact representation.*

Proof sketch: If there are H possible patrol routes $R_1, R_2, ..., R_H$, a mixed defender strategy can be represented in full representation as a probability vector $(p(R_1), ...p(R_H))$ where $p(R_u)$ is the probability of taking route R_u. Taking route R_u means the patroller moves from $d_{r_u(k)}$ to $d_{r_u(k+1)}$ during time $[t_k, t_{k+1}]$, so the edge $E_{R_u(k),R_u(k+1),k}$ is taken when route R_u is chosen. Then the total probability of taking edge $E_{i,j,k}$ is the sum of probabilities of all the routes R_u where $R_u(k) = i$ and $R_u(k+1) = j$. Therefore, given any strategy in full presentation specified by the probability vector $(p(R_1), ...p(R_H))$, we can construct a compact representation consisting of a set of flow distribution variables $\{f(i,j,k)\}$ where

$$f(i,j,k) = \sum_{R_u : R_u(k)=i \text{ and } R_u(k+1)=j} p(R_u). \tag{4}$$

□

Figure 3 shows a simple example illustrating the compact representation. Numbers on the edges indicate the value of $f(i,j,k)$. We use $E_{i,j,k}$ to denote the directed edge linking nodes (t_k, d_i) and (t_{k+1}, d_j). For example, $f(2,1,1)$, the probability of the patroller moving from d_2 to d_1 during time t_1 to t_2, is shown on the edge $E_{2,1,1}$ from node (t_1, d_2) to node (t_2, d_1). While a similar compact representation was used earlier by Yin et al. (2012), we use it in a continuous setting.

Note that different mixed strategies in full representation can be mapped to the same compact representation. Table 1 shows two different mixed defender strategies in full representations that can be mapped to the same mixed strategy in compact representation as shown in Figure 3. The probability of a route is labeled on all edges in the route in full representation. Adding up the numbers of a particular edge $E_{i,j,k}$ in all routes of a full representation together, we can get $f(i,j,k)$ for the compact representation.

Theorem 1. *Compact representation does not lead to any loss in solution quality.*

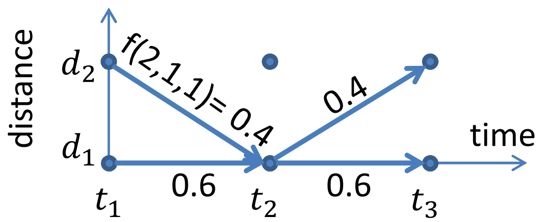


Figure 3: Compact representation: x-axis shows time intervals; y-axis the discretized distance-points in the one-dimensional movement space.

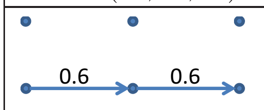
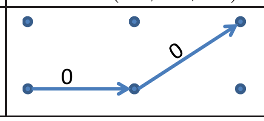
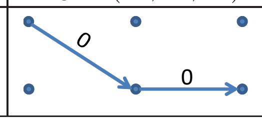
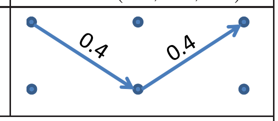
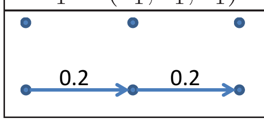
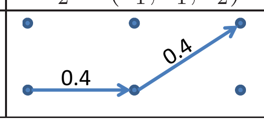
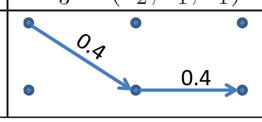
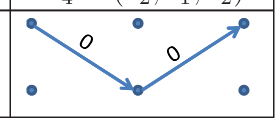
Full Representation 1			
$R_1 = (d_1, d_1, d_1)$	$R_2 = (d_1, d_1, d_2)$	$R_3 = (d_2, d_1, d_1)$	$R_4 = (d_2, d_1, d_2)$
			
Full Representation 2			
$R_1 = (d_1, d_1, d_1)$	$R_2 = (d_1, d_1, d_2)$	$R_3 = (d_2, d_1, d_1)$	$R_4 = (d_2, d_1, d_2)$
			

Table 1: Two full representations that can be mapped into the same compact representation shown in Figure 3.

Proof sketch: The complete proof of the theorem relies on the calculations in Section 3.2 and 3.3. Here we provide a sketch. Recall our goal is to find an optimal defender strategy f that minimizes the maximum attacker expected utility AttEU_f^m. As we will show in the next subsections, $\omega(F_q, t)$ can be calculated from the compact representation $\{f(i,j,k)\}$. If two defender strategies under the full representation are mapped to the same compact representation $\{f(i,j,k)\}$, they will have the same ω function and then the same AttEU function according to Equation 1. Thus the value of AttEU_f^m is the same for the two defender strategies. So an optimal mixed defender strategy in compact representation is still optimal for the corresponding defender strategies in full representation.

We exploit the following properties of the compact representation.

Property 1. *For any time interval $[t_k, t_{k+1}]$, the sum of all flow distribution variables equals to 1: $\sum_{i=1}^N \sum_{j=1}^N f(i,j,k) = 1$.*

Property 2. *The sum of flows that go into a particular node equals the sum of flows that go out of the node. Denote the sum for node (t_k, d_i) by $p(i,k)$, then $p(i,k) = \sum_{j=1}^N f(j,i,k-1) = \sum_{j=1}^N f(i,j,k)$. Each $p(i,k)$ is equal to the marginal probability that the patroller is at location d_i at time t_k.*

Property 3. *Combining Property 1 and 2, $\sum_{i=1}^N p(i,k) = 1$.*

3.2 DASS: Discretized Attacker Strategies

In this subsection, we introduce DASS, a mathematical program that efficiently finds minimax solutions for MRMT_{sg}-based games under the assumption that the attacker will attack at one of the discretized time points t_k. In this problem, we need to minimize v where v is the maximum of attacker's expected utility. Here, v is the maximum of $\text{AttEU}(F_q, t)$ for any target F_q at any discretized time point t_k.

From Equation (1), we know that $\text{AttEU}(F_q, t)$ is decided by $\omega(F_q, t)$, the probability that the patroller is protecting target F_q at time t. Given the position of the target $S_q(t)$, we define the protection range $\beta_q(t) = [\max\{S_q(t)-r_e, d_1\}, \min\{S_q(t)+r_e, d_N\}]$. If the patroller is located within the range $\beta_q(t)$, the distance between the target and the patroller is no more than r_e and thus the patroller is protecting F_q at time t. So $\omega(F_q, t)$ is the probability that the patroller is located within range $\beta_q(t)$ at time t. For the discretized time points t_k, the patroller can only be located at a discretized distance point d_i, so we define the following.

Definition 4. *$I(i, q, k)$ is a function of two values. $I(i, q, k) = 1$ if $d_i \in \beta_q(t_k)$, and otherwise $I(i, q, k) = 0$.*

In other words, $I(i, q, k) = 1$ means that a patroller located at d_i at time t_k can protect target F_q. Note that the value of $I(i, q, k)$ can be calculated directly from the input parameters (d_i, $S_q(t)$ and r_e) and stored in a look-up table. In particular, $I(i, q, k)$ is not a variable in the formulations that follow. It simply encodes the relationship between d_i and the location of target F_q at t_k. The probability that the patroller is at d_i at time t_k is $p(i, k)$. So we have

$$\omega(F_q, t_k) = \sum\nolimits_{i:I(i,q,k)=1} p(i, k), \tag{5}$$

$$\text{AttEU}(F_q, t_k) = \left(1 - C_1 \sum\nolimits_{i:I(i,q,k)=1} p(i, k)\right) U_q(t_k). \tag{6}$$

Equation (6) follows from Equations (1) and (5), expressing attacker's expected utility for discretized time points. Finally, we must address speed restrictions on the patroller. We set all flows corresponding to actions that are not achievable to zero,[1] that is, $f(i, j, k) = 0$ if $|d_j - d_i| > v_m \delta t$. Thus, DASS can be formulated as a linear program. This linear program

1. Besides the speed limit, we can also model other practical restrictions of the domain by placing constraints on $f(i, j, k)$.

solves for any number of targets but only one defender resource.

$$\min_{f(i,j,k),p(i,k)} z \tag{7}$$

$$f(i,j,k) \in [0,1], \qquad \forall i,j,k \tag{8}$$

$$f(i,j,k) = 0, \qquad \forall i,j,k \text{ such that } |d_j - d_i| > v_m \delta t \tag{9}$$

$$p(i,k) = \sum_{j=1}^{N} f(j,i,k-1), \qquad \forall i, \forall k > 1 \tag{10}$$

$$p(i,k) = \sum_{j=1}^{N} f(i,j,k), \qquad \forall i, \forall k < M \tag{11}$$

$$\sum_{i=1}^{N} p(i,k) = 1, \qquad \forall k \tag{12}$$

$$z \geq \text{AttEU}(F_q, t_k), \qquad \forall q, \forall k \tag{13}$$

Constraint 8 describes the probability range. Constraint 9 describes the speed limit. Constraints 10–11 describes Property 2. Constraint 12 is exactly Property 3. Property 1 can be derived from Property 2 and 3, so it is not listed as a constraint. Constraint (13) shows the attacker chooses the strategy that gives him the maximal expected utility among all possible attacks at discretized time points; where AttEU$(\cdot)$ is described by Equation (6).

3.3 CASS: Continuous Attacker Strategies

In this subsection, we generalize the problem to one with continuous attacker strategy set and provides a linear-programming-based solution CASS. CASS efficiently finds optimal mixed defender strategy under the assumption that the attacker can attack at any time in the continuous time interval $T = [0,1]$. With this assumption, DASS's solution quality guarantee may fail: if the attacker chooses to attack between t_k and t_{k+1}, he may get a higher expected reward than attacking at t_k or t_{k+1}. Consider the following example, with the defender's compact strategy between t_k and t_{k+1} shown in Figure 4. Here the defender's strategy has only three non-zero flow variables $f(3,4,k) = 0.3$, $f(3,1,k) = 0.2$, and $f(1,3,k) = 0.5$, indicated by the set of three edges $E^+ = \{E_{3,4,k}, E_{3,1,k}, E_{1,3,k}\}$. A target F_q moves from d_3 to d_2 at constant speed during $[t_k, t_{k+1}]$. Its schedule is depicted by the straight line segment S_q. The dark lines L_q^1 and L_q^2 are parallel to S_q with distance r_e. The area between them indicates the protection range $\beta_q(t)$ for any time $t \in (t_k, t_{k+1})$. Consider the time points at which an edge from E^+ intersects one of L_q^1, L_q^2, and label them as $\theta_{qk}^r, r = 1 \ldots 4$ in Figure 4). Intuitively, these are all the time points at which a defender patrol could potentially enter or leave the protection range of the target. To simplify the notation, we denote t_k as θ_{qk}^0 and t_{k+1} as θ_{qk}^5. For example, a patroller moving from d_3 to d_4 (or equivalently, taking the edge $E_{3,4,k}$) protects the target from θ_{qk}^0 to θ_{qk}^1 because $E_{3,4,k}$ is between L_1^1 and L_1^2 in $[\theta_{qk}^0, \theta_{qk}^1]$, during which the distance to the target is less or equal than protection radius r_e. Consider the sub-intervals between each θ_{qk}^r and θ_{qk}^{r+1}, for $r = 0 \ldots 4$. Since within each of these five sub-intervals, no patroller enters or leaves the protection range, the probability that the target is being protected is a constant in each sub-interval, as shown in Figure 5(a).

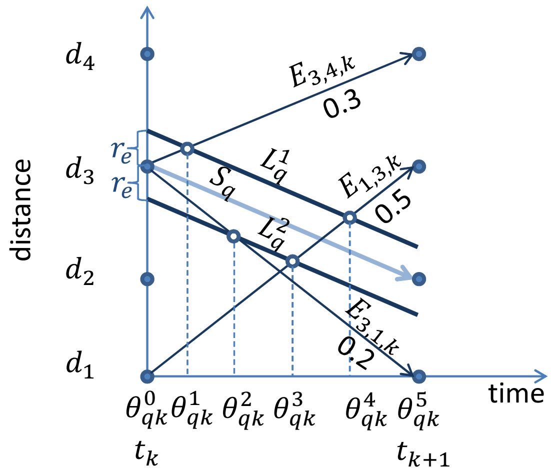


Figure 4: An example to show how a target moving from d_3 to d_2 during $[t_k, t_{k+1}]$ is protected. In a sub-interval $[\theta_{qk}^r, \theta_{qk}^{r+1}]$, a patroller either always protects the target or never protects the target. Equivalently, the target is either always within the protective circle of a patroller or always outside the circle.

Suppose $U_q(t)$ decreases linearly from 2 to 1 during $[t_k, t_{k+1}]$ and $C_1 = 0.8$. We can then calculate the attacker's expected utility function $\text{AttEU}(F_q, t)$ for (t_k, t_{k+1}), as plotted in Figure 5(b). $\text{AttEU}(F_q, t)$ is linear in each sub-interval but the function is discontinuous at the intersection points $\theta_{qk}^1, \ldots, \theta_{qk}^4$ because of the patroller leaving or entering the protection range of the target. We denote the limit of $AttEU$ when t approaches θ_{qk}^r from the left as:

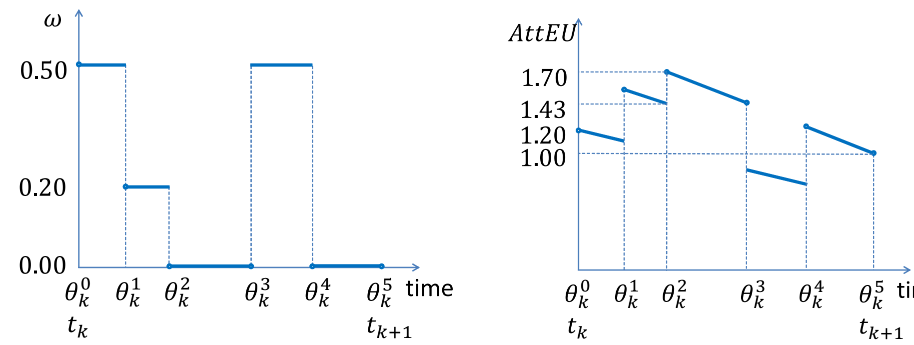
$$\lim_{t \to \theta_{qk}^{r-}} \text{AttEU}(F_q, t) = \text{AttEU}(F_q, \theta_{qk}^{r-})$$

Similarly, the right limit is denoted as:

$$\lim_{t \to \theta_{qk}^{r+}} \text{AttEU}(F_q, t) = \text{AttEU}(F_q, \theta_{qk}^{r+})$$

If F_q is the only target, an attacker can choose to attack at a time immediately after θ_{qk}^2, getting an expected utility that is arbitrarily close to 1.70. According to Equation (6), we can get $\text{AttEU}(F_q, t_k) = 1.20$ and $\text{AttEU}(F_q, t_{k+1}) = 1.00$, both lower than $\text{AttEU}(F_q, \theta_{qk}^{2+})$. Thus, the attacker can get a higher expected reward by attacking between t_k and t_{k+1}, violating DASS's quality guarantee.

However, because of discontinuities in the attacker's expected utility function, a maximum might not exist. This implies that the minimax solution concept might not be well-defined for our game. We thus define our solution concept to be minimizing the *supremum* of $\text{AttEU}(F_q, t)$.



(a) Probability that the target is protected is a constant in each sub-interval.

(b) The attacker's expected utility is linear in each sub-interval.

Figure 5: Sub-interval analysis in (t_k, t_{k+1}) for the example shown in Figure 4.]

Definition 5. *The supremum of attacker's expected utility is the smallest real number that is greater than or equal to all elements of the infinite set* $\{AttEU(F_q, t)\}$*, denoted as* $\sup AttEU(F_q, t)$*.*

The supremum is the least upper bound of the function $\text{AttEU}(F_q, t)$. So for CASS model, Equation 2 should be modified as

$$\text{AttEU}_f^m = \sup_{q,t} \text{AttEU}_f(F_q, t) \tag{14}$$

So a defender strategy f is minimax if AttEU_f^m is maximized, i.e.,

$$f \in \arg\min_{f'} \sup \text{AttEU}_{f'}(F_q, t)$$

In the above example, the supremum of attacker's expected utility in (t_k, t_{k+1}) is $\text{AttEU}(F_q, \theta_{qk}^{2+}) = 1.70$. In the rest of the paper, we will not specify when supremum is used instead of maximum as it can be easily inferred from the context.

How can we deal with the possible attacks between the discretized points and find an optimal defender strategy? We generalize the process above (called sub-interval analysis) to all possible edges $E_{i,j,k}$. We then make use of the piecewise linearity of $\text{AttEU}(F_q, t)$ and the fact that the potential discontinuity points are fixed, which allows us to construct a linear program that solves the problem to **optimality**. We name the approach CASS (Solver for Continuous Attacker Strategies).

We first introduce the general sub-interval analysis. For any target F_q and any time interval (t_k, t_{k+1}), we calculate the time points at which edges $E_{i,j,k}$ and L_q^1, L_q^2 intersect, denoted as intersection points. We sort the intersection points in increasing order, denoted as $\theta_{qk}^r, r = 1 \dots M_{qk}$, where M_{qk} is the total number of intersection points. Set $\theta_{qk}^0 = t_k$ and $\theta_{qk}^{M_{qk}+1} = t_{k+1}$. Thus (t_k, t_{k+1}) is divided into sub-intervals $(\theta_{qk}^r, \theta_{qk}^{r+1}), r = 0, ..., M_{qk}$.

Lemma 1. *For any given target* F_q*,* $AttEU(F_q, t)$ *is piecewise-linear in* t*. Furthermore, there exists a fixed set of time points, independent of the defender's mixed strategy, such that* $AttEU(F_q, t)$ *is linear between each adjacent pair of points. Specifically, these points are the intersection points* θ_{qk}^r *defined above.*

Proof: In each sub-interval $(\theta^r_{qk}, \theta^{r+1}_{qk})$ for a target F_q, a feasible edge $E_{i,j,k}$ is either totally above or below L^1_q, and similarly for L^2_q. Otherwise there will be a new intersection point which contradicts the definition of the sub-intervals. If edge $E_{i,j,k}$ is between L^1_q and L^2_q, the distance between a patroller taking the edge and target F_q is less than r_e, meaning the target is protected by the patroller. As edge $E_{i,j,k}$ is taken with probability $f(i,j,k)$, the total probability that the target is protected ($\omega(F_q,t)$) is the sum of $f(i,j,k)$ whose corresponding edge $E_{i,j,k}$ is between the two lines in a sub-interval. So $\omega(F_q,t)$ is constant in t in each sub-interval and thus the attacker's expected utility $\text{AttEU}(F_q,t)$ is linear in each sub-interval according to Equation 2 as $U_q(t)$ is linear in $[t_k, t_{k+1}]$. Discontinuity can only exist at these intersection points and an upper bound on the number of these points for target F_q is MN^2. □

Define coefficient $A^r_{qk}(i,j)$ to be C_1 if edge $E_{i,j,k}$ is between L^1_q and L^2_q in $(\theta^r_{qk}, \theta^{r+1}_{qk})$, and 0 otherwise. According to Equation (1) and the fact that $\omega(F_q,t)$ is the sum of $f(i,j,k)$ whose corresponding coefficient $A^r_{qk}(i,j) = C_1$, we have the following equation for $t \in (\theta^r_{qk}, \theta^{r+1}_{qk})$.

$$\text{AttEU}(F_q,t) = \left(1 - \sum_{i=1}^{N}\sum_{j=1}^{N} A^r_{qk}(i,j) f(i,j,k)\right) \cdot U_q(t) \tag{15}$$

Piecewise linearity of $\text{AttEU}(F_q,t)$ means the function is monotonic in each sub-interval and the supremum can be found at the intersection points. Because of linearity, the supremum of AttEU in $(\theta^r_{qk}, \theta^{r+1}_{qk})$ can only be chosen from the one-sided limits of the endpoints, $\text{AttEU}(F_q, \theta^{r+}_{qk})$ and $\text{AttEU}(F_q, \theta^{(r+1)-}_{qk})$. Furthermore, if $U_q(t)$ is decreasing in $[t_k, t_{k+1}]$, the supremum is
$\text{AttEU}(F_q, \theta^{r+}_{qk})$ and otherwise it is $\text{AttEU}(F_q, \theta^{(r+1)-}_{qk})$. In other words, all other attacker's strategies in $(\theta^r_{qk}, \theta^{r+1}_{qk})$ are dominated by attacking at time close to θ^r_{qk} or θ^{r+1}_{qk}. Thus, CASS adds new constraints to Constraints 8–13 which consider attacks to occur at $t \in (t_k, t_{k+1})$. We add one constraint for each sub-interval with respect to the possible supremum value in this sub-interval:

$$\min_{f(i,j,k),p(i,k)} z \tag{16}$$

$$\text{subject to constraints } (8 \dots 13)$$

$$z \geq \max\{\text{AttEU}(F_q, \theta^{r+}_{qk}), \text{AttEU}(F_q, \theta^{(r+1)-}_{qk})\} \tag{17}$$

$$\forall k \in \{1 \dots M\}, q \in \{1 \dots L\}, r \in \{0 \dots M_{qk}\}$$

This linear program stands at the core of CASS and we will not differentiate the name for the solver and the name for the linear program in the following. All the linear constraints included by Constraint 17 can be added to CASS using Algorithm 1. The input of the algorithm include targets' schedules $\{S_q\}$, the protection radius r_e, the speed limit v_m, the set of discretized time points $\{t_k\}$ and the set of discretized distance points $\{d_i\}$. Function $\text{CalInt}(L^1_q, L^2_q, v_m)$ in Line 1 returns the list of all intersection time points between all possible edges $E_{i,j,k}$ and the parallel lines L^1_q, L^2_q, with additional points t_k as θ^0_{qk} and t_{k+1} as $\theta^{M_{qk}+1}_{qk}$. Function $\text{CalCoef}(L^1_q, L^2_q, v_m, \theta^r_{qk}, \theta^{r+1}_{qk})$ in Line 1 returns the coefficient matrix A^r_{qk}. A^r_{qk} can be easily decided by checking the status at the midpoint in time. Set

$t_{mid} = (\theta_{qk}^{r} + \theta_{qk}^{r+1})/2$ and denote the patroller's position at t_{mid} when edge $E_{i,j,k}$ is taken as $E_{i,j,t_{mid}}$, thus $A_{qk}^{r}(i,j) = C_1$ if $E_{i,j,t_{mid}} \in \beta_q(t_{mid})$. Lines 1–1 add a constraint with respect to the larger value of $\mathrm{AttEU}(F_q, \theta_{qk}^{r+})$ and $\mathrm{AttEU}(F_q, \theta_{qk}^{(r+1)-})$ to CASS for this sub-interval $(\theta_{qk}^{r}, \theta_{qk}^{r+1})$. It means when the attacker chooses to attack F_q in this sub-interval, his best choice is decided by the larger value of the two side-limits of AttEU in $(\theta_{qk}^{r}, \theta_{qk}^{r+1})$.

Algorithm 1: Add constraints described in Constraint 17

```
Input: S_q, r_e, v_m, {t_k}, {d_i};
for k ← 1, ..., M − 1 do
    for q ← 1, ..., L do
        L_q^1 ← S_q + r_e, L_q^2 ← S_q − r_e;
        θ_qk^0, ..., θ_qk^{M_qk+1} ← CalInt(L_q^1, L_q^2, v_m);
        for r ← 0, ..., M_qk do
            A_qk^r ← CalCoef(L_q^1, L_q^2, v_m, θ_qk^r, θ_qk^{r+1});
            if U_q(t) is decreasing in [t_k, t_{k+1}] then
                add constraint z ≥ AttEU(F_q, θ_qk^{r+})
            end
            else
                add constraint z ≥ AttEU(F_q, θ_qk^{(r+1)−})
            end
        end
    end
end
```

Theorem 2. *CASS computes (in polynomial time) the exact solution (minimax) of the game with discretized defender strategies and continuous attacker strategies.*

Proof: According to Lemma 1, $\mathrm{AttEU}(F_q, t)$ is piecewise linear and discontinuity can only occur at the intersection points θ_{qk}^{r}. These intersection points divide the time space into sub-intervals. Because of piecewise linearity, the supremum of $\mathrm{AttEU}(F_q, t)$ equals to the limit of an endpoint of at least one sub-interval. For any defender's strategy f that is feasible, a feasible z of the linear program 16-17 is no less than any of the limit values at the intersection points according to Constraint 17 and values at the discretized time points t_k according to Constraint 13, and thus v can be any upper bound of $\mathrm{AttEU}(F_q, t)$ for f. As z is minimized in the objective function, z is no greater than the supremum of $\mathrm{AttEU}(F_q, t)$ given any defender strategy f, and further z will be the minimum of the set of supremum corresponding to all defender strategies. Thus we get the optimal defender strategy f.

The total number of variables in the linear program is $O(MN^2)$. The number of constraints represented in Algorithm 1 is $O(MN^2L)$ as the number of intersection points is at most $2(M-1)N^2$ for each target. The number of constraints represented in Constraints 8–13 is $O(MN^2)$. Thus, the linear program computes the solution in polynomial time. □

Corollary 1. *The solution of CASS provides a feasible defender strategy of the original continuous game and gives exact expected value of that strategy.*

3.4 Generalized Model with Multiple Defender Resources

In this subsection, we generalize DASS and CASS to solve the problem with multiple defender resources. When there are multiple patrollers, the patrollers will coordinate with each other. Recall the protection coefficient C_G in Definition 1, a target is better protected when more patrollers are close to it (within radius r_e). So the protection provided to a target is determined when all patrollers' locations are known. Thus it is not sufficient to calculate the probability that an individual edge is taken as in the single patroller case. Under the presence of multiple patrollers, we need a more complex representation to explicitly describe the defender strategy. To illustrate generalization to the multiple defender resources case, we first take two patrollers as an example. If there are two patrollers, the patrol strategy can be represented using flow distribution variables $\{f(i_1, j_1, i_2, j_2, k)\}$. Here the flow distribution variables are defined on the Cartesian product of two duplicated sets of all feasible edges $\{E_{i,j,k}\}$. $f(i_1, j_1, i_2, j_2, k)$ is the joint probability of the first patroller moving from d_{i_1} to d_{j_1} and the second patroller moving from d_{i_2} to d_{i_2} during time t_k to t_{k+1}, i.e., taking edge $E_{i_1,j_1,k}$ and $E_{i_2,j_2,k}$ respectively. The corresponding marginal distribution variable $p(i_1, i_2, k)$ represents for the probability that the first patroller is at d_{i_1} and the second at d_{i_2} at time t_k. Protection coefficients C_1 and C_2 are used when one or two patrollers are protecting the target respectively.

So the attacker's expected utility can be written as

$$\text{AttEU}(F_q, t) = (1 - (C_1 \cdot \omega_1(F_q, t) + C_2 \cdot \omega_2(F_q, t))) \cdot U_q(t)$$

$\omega_1(F_q, t)$ is the probability that only one patroller is protecting the target F_q at time t and $\omega_2(F_q, t)$ is the probability that both patrollers are protecting the target. For attacks that happen at discretized points t_k, we can make use of $I(i, q, k)$ in Definition 4 and $I(i_1, q, k) + I(i_2, q, k)$ is the total number of patrollers protecting the ferry at time t_k.

$$\omega_1(F_q, t_k) = \sum\nolimits_{i_1, i_2 : I(i_1,q,k)+I(i_2,q,k)=1} p(i_1, i_2, k)$$
$$\omega_2(F_q, t_k) = \sum\nolimits_{i_1, i_2 : I(i_1,q,k)+I(i_2,q,k)=2} p(i_1, i_2, k)$$

Constraints for attacks occurring in (t_k, t_{k+1}) can be calculated with an algorithm that looks the same as Algorithm 1. The main difference is in the coefficient matrix A^r_{qk} and the expression of AttEU. We set the values in the coefficient matrix $A^r_{qk}(i_1, j_1, i_2, j_2)$ as C_2 if both edges $E_{i_1,j_1,k}$ and $E_{i_2,j_2,k}$ are between L^1_q and L^2_q, and C_1 if only one of the edges protects the target. The attacker's expected utility function in $(\theta^r_{qk}, \theta^{r+1}_{qk})$ is

$$\text{AttEU}(F_q, t) = (1 - \sum_{i_1, j_1, i_2, j_2} A^r_{qk}(i_1, j_1, i_2, j_2) f(i_1, j_1, i_2, j_2, k)) \cdot U_q(t)$$

For a general case of W defender resources, we can use $\{f(i_1, j_1, ..., i_W, j_W, k)\}$ to represent the patrol strategy.

Definition 6. *The* compact representation *for multiple defender resources is a compact way to represent the defender's mixed strategy using* flow distribution variables $\{f(i_1, j_1, ..., i_W, j_W, k)\}$. $\{f(i_1, j_1, ..., i_W, j_W, k)\}$ *is the joint probability that patroller moving from* d_{i_u} *at time* t_k *to* d_{j_u} *at time* t_{k+1} *for* $u = 1 \ldots W$.

Given the generalized compact representation, we get the following equations for calculating the attacker's expected utility function and the protection probability:

$$\mathrm{AttEU}(F_q,t) = \left(1 - \sum_{Q=1}^{W} C_Q \cdot \omega_Q(F_q,t)\right) \cdot U_q(t)$$
$$\omega_Q(F_q,t_k) = \sum\nolimits_{i_1,\ldots,i_W:\sum\limits_{u=1}^{W} I(i_u,q,k)=Q} p(i_1,\ldots,i_W,k)$$

Q is the number of patrollers protecting the target. We can modify Algorithm 1 to apply for the multiple defender resource case. Set $A^r_{qk}(i_1, j_1, ..., i_W, j_W)$ as C_Q if Q of the edges $\{E_{i_u,j_u,k}\}$ are between L^1_q and L^2_q.

We conclude the linear program for generalized CASS for multiple patrollers as follows.

$$\min_{f(i_1,j_1,\ldots,i_W,j_W,k),p(i_1,\ldots,i_W,k)} z \tag{18}$$

$$f(i_1,j_1,\ldots,i_W,j_W,k) = 0, \forall i_1,\ldots,i_W,j_1,\ldots,j_W \text{ such that } \exists u, |d_{j_u} - d_{i_u}| > v_m\delta t \tag{19}$$

$$p(i_1,\ldots,i_W,k) = \sum_{j_1=1}^{n}\cdots\sum_{j_W=1}^{n} f(j_1,i_1,\ldots,j_W,i_W,k-1), \forall i_1,\ldots,i_W, \forall k>1 \tag{20}$$

$$p(i_1,\ldots,i_W,k) = \sum_{j_1=1}^{n}\cdots\sum_{j_W=1}^{n} f(i_1,j_1,\ldots,i_W,j_W,k), \forall i_1,\ldots,i_W, \forall k<M \tag{21}$$

$$\sum_{i_1=1}^{n}\cdots\sum_{i_W=1}^{n} p(i_1,\ldots,i_W,k) = 1, \forall k \tag{22}$$

$$z \geq \mathrm{AttEU}(F_q,t_k), \forall q, \forall k \tag{23}$$

$$z \geq \max\{\mathrm{AttEU}(F_q,\theta^{r+}_{qk}), \mathrm{AttEU}(F_q,\theta^{(r+1)-}_{qk})\}, \forall k, \forall q, \forall r \tag{24}$$

The number of variables in the linear program is $O(MN^{2W})$ and the number of constraints is $O(MN^W)$. It is reasonable to examine potentially more efficient alternatives. We summarize the results of such an examination below concluding that using the current linear program would appear to currently offer our best tradeoff in terms of solution quality and time at least for the current domains of application; although as discussed below, significant future work might reveal alternatives approaches for other future domains.

The first question to examine is that of the computational complexity of the problem at hand: generating optimal patrolling strategies for multiple patrollers on a graph. Unfortunately, despite significant attention paid to the topic, currently, the complexity remains unknown. More specifically, the question of computational complexity of generating patrols for multiple defenders on graphs of different types has received significant attention (Letchford, 2013; Korzhyk et al., 2010). These studies illustrate that in several cases the problem is NP-hard, in some cases the problem is known to be polynomial time, but despite significant effort, the problem complexity in many cases remains unknown (Letchford & Conitzer, 2013). Unfortunately, our graph turns out to be different from the cases considered in their work. Indeed, the DASS model can be explained as a game with homogeneous defender resources patrolling on a graph, similar to the cases that have already been considered. However, prior results cannot explain the complexity of our game as the structure of our graph does not fit any of the prior graphs.

Given that computational complexity results are not directly available, we may examine approaches to provide efficient approximations. Here we provide an overview of two such approaches (providing experimental results in Section 7.1.6). Our first approach attempts to provide a more compact representation in the hope of providing a speedup. To that end, we apply an intuitive approach that uses individual strategy profile for each patroller and then calculates a best possible mixed strategy combination. Unfortunately, this approach is inefficient in run-time even for the DASS model and may result in a suboptimal solution. Thus, although more compact, this approach fails to achieve our goal; we explain this approach next.

Assume each patroller independently follows her own mixed strategy. Denote the individual mixed strategy for patroller u as $f_u(i_u, j_u, t_k)$, and the probability that a target is protected by Q players can be represented as a polynomial expression of $\{f_u(i_u, j_u, t_k)\}$ of order Q. Then our optimization problem is converted to minimizing objective function z with non-linear constraints. Assume we have two patrollers, and for a potential attack at target q at time t_k, we denote the probability that patroller u is protecting the target as ϖ_u. ϖ_u is linear in f_u, and the attacker's expected utility for this attack can be represented as

$$\mathrm{AttEU}(F_q, t_k) = (1 - C_1((1-\varpi_1)\varpi_2 + (1-\varpi_2)\varpi_1) - C_2\varpi_1\varpi_2)U_q(t_k)$$

So a constraint $z \geq \mathrm{AttEU}(F_q, t_k)$ is quadratic in f, due to the fact that the joint probability is represented by the product of the individual probability of each patroller. These constraints are not ensured to have a convex feasible region and there is no known polynomial algorithms for solving this kind of non-convex optimization problems. We attempt to solve the problem by converting it into a mathematical program with a non-convex objective function and linear constraints, i.e., instead of minimizing z with constraints $z \geq \mathrm{AttEU}(F_q, t_k)$, we incorporate the constraints into the objective function as

$$z = \max_{q,k}\{\mathrm{AttEU}(F_q, t_k)\} \tag{25}$$

The results in Section 7.1.6 show that when we solve this mathematical program in MATLAB using function **fmincon** with interior-point method for the DASS model, the algorithm fails to get to a feasible solution efficiently and even when enough time is given, the solution can still be suboptimal as it may get stuck at a local minimum. To conclude, although this approach is more compact and helps in saving memory, it is inefficient in run-time and may result in loss in solution quality.

Our second approach takes a further step to reduce the run-time complexity, making it a polynomial approximation algorithm, but it can lead to a high degradation in solution quality. In this approach, we iteratively compute the optimal defender strategy for a newly added resource unit given the existing strategies for the previous defender resources. Namely, we first calculate $f_1(i_1, j_1, t_k)$ as if only one patroller is available and then calculate

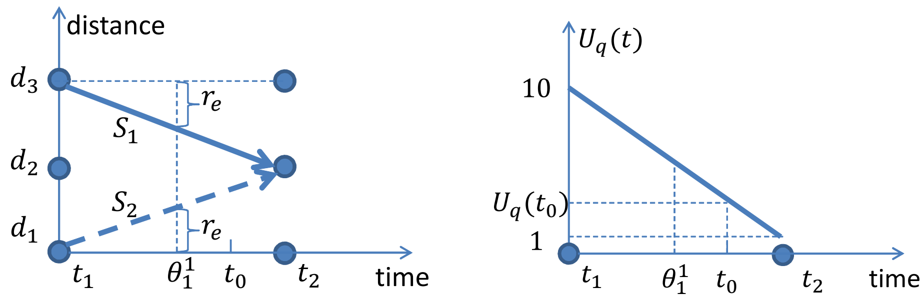
$f_2(i_2, j_2, t_k)$ given the value of $f_1(i_1, j_1, t_k)$. In this way, we need to solve W linear programs with complexity $O(MN^2)$ so this approach is much faster compared to the former one. Unfortunately, this approach fails to capture the coordination between the patrollers effectively and thus may result in a high degradation in solution quality. For example, suppose there are only two targets of constant utility U, one target stays at terminal A and the other one stays at terminal B. Further, suppose the protection coefficient is always 1 when a target is protected by one or more patrollers. When two patrollers are available, the optimal solution would be each protect one of the targets all the way, so both targets are protected with probability 1 and the expected utility function for the attacker is 0. If the defender strategy is calculated for each patroller sequentially as discussed above, the solution would be protect each target with probability 0.5 for both players, making the attacker's expected utility $0.25\%U$. In other words, we reach a suboptimal solution, wasting resources when both patrollers end up protecting the same target with probability 0.25. In this case, we can already see that there is a 0.25 probability that a target is unprotected when clearly an optimal solution existed that protected all targets with probability 1. Thus, even with just two patrollers this solution leads to a potentially significant loss in expected utility; therefore, this solution clearly appears to be inadequate for our purposes.

Given the above discussion, it would appear that a fast approximation may lead to significant losses in solution quality or may not be efficient enough. Fortunately for current application domains, such as the current deployment of CASS for protecting ferries (e.g., the Staten Island Ferry in New York), the number of defender resources are limited. The lack of resources is the main reason that optimization using security games becomes critical. As a result, our current approach of CASS is adequate for current domains such as ferry protection. Further research about scale-up is an issue for future work.

4. Equilibrium Refinement

A game often has multiple equilibria. Since our game is zero-sum, all equilibria achieve the same objective value. However, if an attacker deviates from his best response, some equilibrium strategies for the defender may provide better results than others.

Consider the following example game. There are two targets moving during $[t_1, t_2]$ (no further discretization): one moves from d_3 to d_2 and the other moves from d_1 to d_2 (See Figure 6(a)). Suppose $d_3 - d_2 = d_2 - d_1 = \Delta d$ and $r_e = 0.5\Delta d$. There is only one patroller available and the protection coefficient $C_1 = 1$. Both targets' utility functions decrease from 10 to 1 in $[t_1, t_2]$ (See Figure 6(b)). In one equilibrium, $f_{3,2,1} = f_{1,2,1} = 0.5$, i.e., the patroller randomly chooses one target and follows it all the way. In another equilibrium, $f_{3,3,1} = f_{1,1,1} = 0.5$, i.e., the patroller either stays at d_1 or at d_3. In either equilibrium, the attacker's best response is to attack at t_1, with a maximum expected utility of 5. However, if an attacker is physically constrained (e.g., due to launch point locations) to only attack no earlier than t_0 and $t_0 > \theta_1^1$ (where θ_1^1 is the only intersection time point and $\theta_1^1 = (t_1 + t_2)/2$), against both defender strategies he will choose to attack either of the targets at t_0. The attacker's expected utility is $U_q(t_0)/2$ in the first equilibrium because there is 50% probability that the patroller is following that target. However in the second equilibrium, he is assured to succeed and get a utility of $U_q(t_0)$ because the distance between the chosen target and d_1 (or d_3) is larger than r_e at t_0, i.e., the chosen target is unprotected at t_0. In this case, the defender strategy in the first equilibrium is preferable to the one in the second; indeed, the first defender strategy dominates the second one, by which we mean the first is equally good or better than the second no matter what strategy the attacker chooses. We provide a formal definition of dominance in Section 4.1.



(a) Two targets moves with schedules S_1 and S_2.

(b) Utility function is the same for both targets and is decreasing linearly over time.

Figure 6: An example to show one equilibrium outperforms another when the attacker is constrained to attack in $[t_0, t_2]$ if $t_0 > \theta_1^1$.

Our goal is to improve the defender strategy so that it is more robust against constrained attackers while keeping the defender's expected utility against unconstrained attackers the same. This task of selecting one from the multiple equilibria of a game is an instance of the *equilibrium refinement* problem, which has received extensive study in game theory (van Damme, 1987; Fudenberg & Tirole, 1991; Miltersen & Sørensen, 2007). For finite security games, An, Tambe, Ordóñez, Shieh, and Kiekintveld (2011) proposed techniques that provide refinement over Stackelberg equilibrium. However there has been little prior research on the computation of equilibrium refinements for continuous games.

In this section, we introduce two equilibrium refinement approaches: "route-adjust" (Section 4.1) and "flow-adjust" (Section 4.2). Both approaches can be applied to improve any feasible defender strategy and when they are applied to an optimal defender strategy in an existing equilibrium, we will get new equilibria with more robust optimal defender strategies.

For expository simplicity, we still use the single-resource case as an example, but both methods are applicable to the multiple-resources case. The results shown in evaluation section experimentally illustrates these two refinement methods can significantly improve the performance.

4.1 Route Adjust

Given that f is the defender strategy of one equilibrium of the game, if we can find a defender strategy f' such that for any attacker strategy (q, t), the defender's expected utility under f' is equal to or higher than the one under f, and the one under f' is strictly higher than the one under f for at least one specific attacker strategy, we say that f' dominates f. Intuitively, the defender should choose f' instead of f as f' is at least as good as f for any attacker strategy and can achieve better performance for some attacker strategies. So an equilibrium with strategy f' is more robust to unknown deviations on the attacker side. We give the formal definition of dominance as follows.

Definition 7. *Defender strategy f **dominates** f' if $\forall q, t$, $DefEU_f(F_q, t) \geq DefEU_{f'}(F_q, t)$, and $\exists q, t$, $DefEU_f(F_q, t) > DefEU_{f'}(F_q, t)$; or equivalently in this zero-sum game, $\forall q, t$, $AttEU_f(F_q, t) \leq AttEU_{f'}(F_q, t)$, and $\exists q, t$, $AttEU_f(F_q, t) < AttEU_{f'}(F_q, t)$.*

Corollary 2. *Defender strategy f **dominates** f' if $\forall q, t$, $\omega(F_q, t) \geq \omega'(F_q, t)$ and $\exists q, t$, $\omega(F_q, t) > \omega'(F_q, t)$.*

Definition 7 simply restates the commonly used weak dominance definition in game theory for this specific game. Corollary 2 follows from Equation (1).

In this section, we introduce the route-adjust approach which gives a procedure for finding a defender strategy f^1 that dominates the given defender strategy f^0. Route-adjust provides final routes using these steps: (i) decompose flow distribution f^0 into component routes; (ii) for each route, greedily find a route which provides better protection to targets; (iii) combine the resulting routes into a new flow distribution, f^1, which dominates f^0 if f^1 is different from f_0. The detailed process is listed in Algorithm 2. We illustrate this approach using a simple dominated strategy shown in Figure 3.

To accomplish step (i), we decompose the flow distribution by iteratively finding a route that contains the edge with minimum probability. As shown in Figure 7, we first randomly choose a route that contains edge $E_{1,2,2}$, as $f(1, 2, 2) = 0.4$ is the minimum among all flow variables. We choose $R_2 = (d_1, d_1, d_2)$, and set $p(R_2) = f(1, 2, 2) = 0.4$. Then for each edge of the route R_2 we subtract 0.4 from the original flow, resulting in a residual flow. We continue to extract routes from the residual flow until there is no route left. Denote by Z the number of non-zero edges in the flow distribution graph, then Z is decreased by at least 1 after each iteration. So the algorithm will terminate in at most Z steps and at most Z routes are found. The result of step (i) is a sparse description of a defender mixed strategy in full representation. As we will discuss in Section 6, this decomposition constitutes one method of executing a compact strategy.

For step (ii), we adjust each of the routes greedily. To that end, we first introduce the dominance relation of edges and routes, using the intersection points θ^r_{qk} and the coefficient matrix $A^r_{qk}(i, j)$ defined in Section 3.3.

Definition 8. *Edge $E_{i,j,k}$ **dominates** edge $E_{i',j',k}$ in $[t_k, t_{k+1}]$ if $A^r_{qk}(i, j) \geq A^r_{qk}(i', j')$, $\forall q = 1..L$, $\forall r = 0..M_{qk}$, and $\exists q, r$ such that $A^r_{qk}(i, j) > A^r_{qk}(i', j')$.*

The dominance relation of edges is based on the comparison of protection provided to the targets in each sub-interval. In the following dominance relation of routes, we denote the edge $E_{r_u(k), r_u(k+1), k}$ as $E(u, k)$ to simplify the notation, .

Definition 9. *Route $R_u = (d_{r_u(1)}, \ldots, d_{r_u(M)})$ **dominates** $R_{u'} = (d_{r_{u'}(1)}, \ldots, d_{r_{u'}(M)})$ if $\forall k = 1 \ldots M-1$, $E(u, k) = E(u', k)$ or $E(u, k)$ dominates $E(u', k)$ and $\exists k$ such that $E(u, k)$ dominates $E(u', k)$.*

Route R_u dominates $R_{u'}$ if each edge of R_u is either the same as or dominates the corresponding edge in $R_{u'}$ and at least one edge in R_u dominates the corresponding edge in $R_{u'}$.

Algorithm 2: Route-Adjust

Input: a mixed defender strategy f

Output: an updated mixed defender strategy f'

(i) Decompose f into multiple routes by iteratively finding a route that contains the edge with minimum probability:

(a) Initialize the remaining flow distribution $\tilde{f} = f$ and route set $S = \emptyset$. Initialize probability distribution over routes $p(R_u) = 0$, $\forall u$.

(b) **while** $\max \tilde{f}(i, j, k) > 0$ **do**

i. Set $(i_0, j_0, k_0) = \arg\min_{i,j,k:\tilde{f}(i,j,k)>0} \tilde{f}(i, j, k)$.

ii. Set $f_{min} = \tilde{f}(i_0, j_0, k_0)$.

iii. Find an arbitrary route R_{u_0} such that $r_{u_0}(k_0-1) = i_0$ and $r_{u_0}(k_0) = j_0$ (i.e., edge E_{i_0,j_0,k_0} is in the route) and $\tilde{f}(r_{u_0}(k), r_{u_0}(k+1), k) > 0$, $\forall k$ (i.e., all edges in the route has non-zero remaining flow).

iv. Add R_{u_0} to S and set $p(R_{u_0}) = f_{min}$.

v. Set $\tilde{f}(i, j, k) = \tilde{f}(i, j, k) - f_{min}$ if $r_{u_0}(k-1) = i$ and $r_{u_0}(k) = j$.

end

(ii) Adjust each route in S greedily to get a new set of routes S' and the corresponding new probability distribution p':

(a) Initialize the new set $S' = \emptyset$ and new probability distribution $p'(R_u) = 0$, $\forall u$.

(b) **while** $S \neq \emptyset$ **do**

i. Pick a route R_u from S.

ii. Adjust R_u to get new route $R_{u'}$: for a given R_u and a specified k^*, set $r_{u'}(k) = r_u(k)$ if $k \neq k^*$. Set $r_{u'}(k^*) = i_0$ such that: 1) $E(u_1, k^* - 1)$ and $E(u_1, k^*)$ meet the speed constraint; 2) $R_{u'}$ dominates R_u with the choice of i_0; 3) $R_{u'}$ is not dominated by a route with any other choice of i_0. If no such i_0 exists, set $r_{u'}(k^*) = r_u(k^*)$

iii. Add R_u to S' and set $p'(R_{u'}) = p(R_u)$.

iv. Remove R_u from S.

end

(iii) Reconstruct a new compact representation f' from S' and p' according to Equation 4.

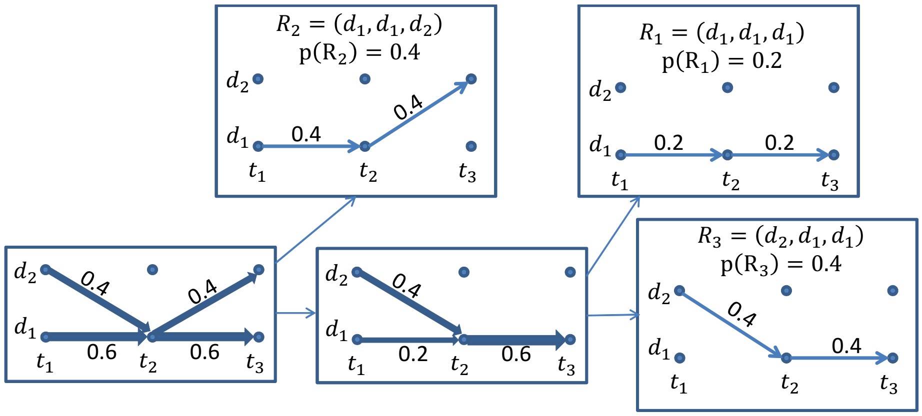


Figure 7: Step (i): decomposition. Every time a route containing the minimal flow variable is subtracted and a residual graph is left for further decomposition. The original flow distribution is thus decomposed into three routes R_2, R_1, and R_3 with probability 0.4, 0.2 and 0.4 respectively.

Denote the original route to be adjusted as R_u and the new route as $R_{u'}$. A greedy way to improve the route is to replace only one node in the route. If we want to replace the node at time t_{k^*}, then we have $r_{u'}(k) = r_u(k)$, $\forall k \neq k^*$ and $d_{r_u(k^*)}$ in the original route is replaced with $d_{r_{u'}(k^*)}$. So the patroller's route changes only in $[t_{k^*-1}, t_{k^*+1}]$. Thus, only edges $E(u, k^*-1)$ and $E(u, k^*)$ in the original route are replaced by $E(u', k^*-1)$ and $E(u', k^*)$ in the new route.

We are trying to find a new route $R_{u'}$ that dominates the original route to provide equal or more protection to the targets. So the selection of $r_{u'}(k^*)$ needs to meet the requirements specified in Algorithm 2. The first one describes the speed limit constraint. The second one actually requires the changed edges $E(u', k^*-1)$ and $E(u', k^*)$ are either equal to or dominate the corresponding edges in the original route (and dominance relation exist for at least one edge). The third requirement attains a local maximum. If such a new node does not exist for a specified k^*, we return the original route R_u.

We can iterate this process for the new route and get a final route denoted by $R_{u'}$ after several iterations or when the state of convergence is reached. When the state of convergence is reached, the resulting route $R_{u'}$ keeps unchanged no matter which k^* is chosen for the next iteration.

For the example in Figure 7, assume the only target's moving schedule is $d_1 \to d_1 \to d_2$, $d_3 - d_2 = d_2 - d_1 = \Delta d$, $r_e = 0.1\Delta d$ and utility function is constant. We adjust each route for only one iteration by changing the patroller's position at time t_3, i.e., $r_u(3)$. As t_3 is the last discretized time point, only edge $E(u, 2)$ may be changed. For $R_1 = (d_1, d_1, d_1)$, we enumerate all possible patroller's positions at time t_3 and choose one according to the three constraints mentioned above. In this case, the candidates are d_1 and d_2, so the corresponding new routes are R_1 (unchanged) and $R_2 = (d_1, d_1, d_2)$ respectively. Note that edge $E_{d_1,d_2,2}$ dominates $E_{d_1,d_1,2}$ because the former one protects the target all the way in $[t_2, t_3]$ and thus R_2 dominates R_1. So d_2 is chosen as the patroller's position at t_3 and R_2

is chosen as the new route. The adjustment for all routes with non-zero probability after decomposition is shown in Table 2.

R_u	$p(R_u)$ after decomposition	Adjusted Routes
$R_1 = (d_1, d_1, d_1)$	0.2	$(d_1, d_1, d_2) = R_2$
$R_2 = (d_1, d_1, d_2)$	0.4	$(d_1, d_1, d_2) = R_2$
$R_3 = (d_2, d_1, d_1)$	0.4	$(d_2, d_1, d_2) = R_4$

Table 2: Step (ii): Adjust each route greedily.

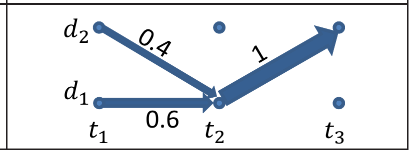
R_u	$p'(R_u)$ after adjustment		Composed Flow Distribution
$R_1 = (d_1, d_1, d_1)$	0		
$R_2 = (d_1, d_1, d_2)$	0.6		
$R_3 = (d_2, d_1, d_1)$	0		
$R_4 = (d_2, d_1, d_2)$	0.4		

Table 3: Step (iii): compose a new compact representation.

The new routes we get after step (ii) are same as the original routes or dominate the original routes. That is, whenever a route R_u is chosen according to the defender mixed strategy resulting from step (i), it is always equally good or better to choose the corresponding new route $R_{u'}$ instead, because $R_{u'}$ provides equal or more protection to the targets than R_u. Suppose there are H possible routes in the defender strategy after step (i), denoted as $R_1, ..., R_H$. After adjusting the routes, we get a new defender strategy $(p'(R_1), p'(R_2), ..., p'(R_H))$ in full representation (See Table 3). Some routes are taken with higher probability (e.g. $p'(R_2) = 0.2 + 0.4 = 0.6$) and some are with lower probability (e.g. $p'(R_3) = 0$) compared to the original strategy. For step (iii), we reconstruct a new compact representation according to Equation 4. This is accomplished via a process that is the inverse of decomposition and is exactly the same as how we map a strategy in full representation into compact representation. For the example above, the result is shown in Table 3.

Theorem 3. *After steps (i)–(iii), we get a new defender strategy f^1 that dominates the original one f^0 if f^1 is different from f^0.*

Proof: We continue to use the notation that the decomposition in step (i) yields the routes $R_1, ..., R_H$. For each flow distribution variable in the original distribution $f^0(i, j, k)$, it is decomposed into H sub-flows $\{f_u^0(i, j, k)\}$ according to the route decomposition. $f_u^0(i, j, k) = p(R_u)$ if $i = r_u(k), j = r_u(k+1)$ and $f_u^0(i, j, k) = 0$ otherwise. Thus we have the following equation.

$$f^0(i, j, k) = \sum\nolimits_{u=1}^{H} f_u^0(i, j, k) \tag{26}$$

$$= \sum\nolimits_{u: r_u(k)=i, r_u(k+1)=j} f_u^0(i, j, k) \tag{27}$$

After adjust each route separately, each non-zero sub-flow $f_u^0(i, j, k)$ on edge $E(u, k)$ is moved to edge $E(u', k)$ as route R_u is adjusted to $R_{u'}$. Reconstructing the flow distribution f^1

can also be regarded as adding up all the sub-flows after adjustment together on each edge. That means, f^1 is composed of a set of sub-flows after adjustment, denoted as $\{f^1_u(i', j', k)\}$. The subscript u represents for the index of the **original** route to indicate it is moved from edge $E(u, k)$. So $f^1_u(i', j', k) = f^0_u(r_u(k), r_u(k + 1), k)$, if $i' = R_{u'}(k)$ and $j' = R_{u'}(k + 1)$; otherwise $f^1_u(i', j', k) = 0$. Similarly to Equation 27, we have the following equation for f^1.

$$f^1(i', j', k) = \sum\nolimits_{u=1}^{H} f^1_u(i', j', k) \tag{28}$$

$$= \sum\nolimits_{u': r_{u'}(k)=i', r_{u'}(k+1)=j'} f^1_u(i', j', k) \tag{29}$$

Based on how the adjustment is made, $R_{u'}$ is same as or dominates R_u and thus $E(u', k)$ is same as or dominates $E(u, k)$. So if edge $E(u, k)$ protects target F_q at time t, the corresponding edge $E(u', k)$ after adjustment also protects target F_q at time t.

Recall from Section 3.3 that $\omega(F_q, t)$ is the sum of $f(i, j, k)$ whose corresponding edge $E_{i,j,k}$ can protect the target F_q at time t. We denote by $\omega^0(F_q, t)$ and $\omega^1(F_q, t)$ the probabilities of protection corresponding to f^0 and f^1 respectively. According to Equation 27, $\omega^0(F_q, t)$ can be viewed as the sum of all the non-zero sub-flows $f^0_u(i, j, k)$ where the corresponding $E(u, k)$ protects the target F_q at time t. If $f^0_u(i, j, k)$ is a term in the summation to calculate $\omega^0(F_q, t)$, it means $E(u, k)$ protects F_q at t and thus the corresponding $E(u', k)$ protects F_q at t, so the corresponding sub-flow $f^1_u(r_{u'}(k), r_{u'}(k+1), k)$ in f^1 is also a term in the summation to calculate $\omega^1(F_q, t)$. It leads to the conclusion $\omega^0(F_q, t) \leq \omega^1(F_q, t)$. Note that if $\forall q, t$, $\omega^0(F_q, t) = \omega^1(F_q, t)$, then all routes kept unchanged in step (ii) as otherwise it contradicts with the fact that the new route dominates the original route. According to Corollary 2, we have f^1 dominates f^0 if it is different from f^0. $\square$

In the example in Figure 7, $f^0(1, 1, 2)$ is decomposed into two non-zero terms $f^0_1(1, 1, 2) = 0.2$ and $f^0_3(1, 1, 2) = 0.4$ along with routes R_1 and R_3 (See Figure 7). After adjustment, we get the corresponding subflows $f^1_1(1, 2, 2) = 0.2$, $f^1_3(1, 2, 2) = 0.4$. Recall that the target's schedule is $d_1 \to d_1 \to d_2$. The flow distribution after adjustment (See Table 5) gives more protection to the target in $[t_2, t_3]$. Since the flow is equal from t_1 to t_2 (and therefore the protection is the same), overall the new strategy dominates the old strategy.

Therefore, if we apply route-adjust to the optimal defender strategy calculated by CASS we get a more robust equilibrium. While step (iii) allows us to prove Theorem 3, notice that at the end of step (ii), we have a probability distribution over a set of routes from which we can *sample actual patrol routes.* For two or more defender resources, a generalized version of Definition 8 can be used to define the dominance relation on the edge tuple $(E_{i_1,j_1,k}, ..., E_{i_W,j_W,k})$ with coefficient matrix for multiple patrollers $A^r_{qk}(i_1, j_1, ..., i_W, j_W)$.

There are other ways to adjust each route. Instead of adjusting only one node in the route, we can adjust more consecutive nodes at a time, for example, we can adjust both $r_{u'}(k^*)$ and $r_{u'}(k^*+1)$ by checking edges $E(u', k^*-1)$, $E(u', k^*)$ and $E(u', k^*+1)$. However, we need to tradeoff the performance and the efficiency of the algorithm. This tradeoff will be further discussed in Section 7.

4.2 Flow Adjust

Whereas route-adjust tries to select an equilibrium that is robust against attackers playing suboptimal strategies, the second approach, *flow-adjust*, attempts to select a new equilibrium that is robust to rational attackers that are constrained to attack during any time interval $[t_k, t_{k+1}]$. As we will discuss below, flow-adjust focuses on a weaker form of dominance, which implies that a larger set of strategies are now dominated (and thus could potentially be eliminated) compared to the standard notion of dominance used by route-adjust; however flow-adjust does not guarantee the elimination of all such dominated strategies. We denote by DefEU_f^k the defender expected utility when an attacker is constrained to attack during time interval $[t_k, t_{k+1}]$ when the attacker provides his best response given the defender strategy f. Formally, $\text{DefEU}_f^k = \min_{q\in\{1\ldots L\}, t\in[t_k,t_{k+1}]}\{\text{DefEU}_f(F_q, t)\}$. We give the following definition of "local dominance".

Definition 10. *Defender strategy f **locally dominates** f' if $DefEU_f^k \geq DefEU_{f'}^k$, $\forall k$.*[2]

Corollary 3. *Defender strategy f **locally dominates** f' if*

$$\min_{q\in\{1\ldots L\}, t\in[t_k,t_{k+1}]}\{DefEU_f(F_q, t)\} \geq \min_{q\in\{1\ldots L\}, t\in[t_k,t_{k+1}]}\{DefEU_{f'}(F_q, t)\}, \forall k,$$

or equivalently in this zero-sum game,

$$\max_{q\in\{1\ldots L\}, t\in[t_k,t_{k+1}]}\{AttEU_f(F_q, t)\} \leq \max_{q\in\{1\ldots L\}, t\in[t_k,t_{k+1}]}\{AttEU_{f'}(F_q, t)\}, \forall k.$$

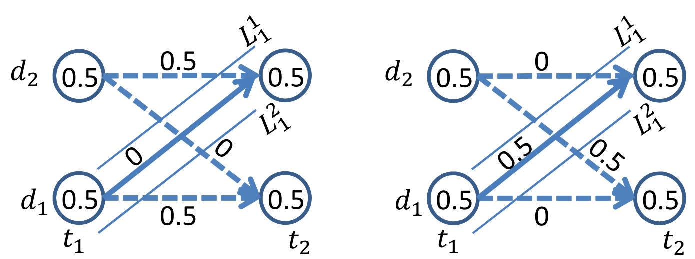
Corollary 3 follows from the fact that the attacker plays a best response given the defender strategy, and it means that f locally dominates f' if the maximum of attacker expected utilities in each time interval $[t_k, t_{k+1}]$ given f is no greater than that of f'.

Compared to Definition 7, which gives the standard condition for dominance, local dominance is a weaker condition; that is, if f dominates f' then f locally dominates f', however the converse is not necessarily true. Intuitively, whereas in Definition 7 the attacker can play any (possibly suboptimal) strategy, here the attacker's possible deviations from best response are more restricted. As a result, the set of locally dominated strategies includes the set of dominated strategies. From Definition 10, if f locally dominates f', and the attacker is rational (i.e., still playing a best response) but constrained to attack during some time interval $[t_k, t_{k+1}]$, then f is preferable to f' for the defender. A further corollary is that even if the rational attacker is constrained to attack in the union of some of these intervals, f is still preferable to f' if f locally dominates f'. One intuition for the local dominance concept is the following: suppose we suspect the attacker will be restricted to a (unknown) subset of time, due to some logistical constraints. Such logistical constraints would likely make the restricted time subset to be contiguous or a union of a small number of contiguous sets. Since such sets are well-approximated by unions of intervals $[t_k, t_k + 1]$, local dominance can serve as an approximate notion of dominance with respect to such attackers.

Flow-adjust looks for a defender strategy f^1 that locally dominates the original defender strategy f^0. To achieve this, we simply adjust the flow distribution variables $f(i, j, k)$ while keeping the marginal probabilities $p(i, k)$ the same. Figure 8 shows an example game with two discretized intervals $[t_1, t_2]$ and $[t_2, t_3]$ (only the first interval is shown). Suppose the maximal attacker expected utility is $5U_0$ in this equilibrium and is attained in the second

2. We don't require that there exists at least one k such that $\text{DefEU}_f^k > \text{DefEU}_{f'}^k$.

interval $[t_2, t_3]$. If the attacker's utility for success is a constant U_0 in the first interval $[t_1, t_2]$, then the defender strategy in $[t_1, t_2]$ could be arbitrarily chosen because the attacker's expected utility in $[t_1, t_2]$ in worst case is smaller than that of the attacker's best response in $[t_2, t_3]$. However, if a attacker is constrained to attack in $[t_1, t_2]$ only, the defender strategy in the first interval will make a difference. In this example, there is only one target moving from d_1 to d_2 during $[t_1, t_2]$. The schedule of the ferry is shown as dark lines and the parallel lines L_1^1 and L_1^2 with respect to protection radius $r_e = 0.2(d_2 - d_1)$ are shown as dashed lines. The marginal distribution probabilities $p(i, k)$ are all 0.5 and protection coefficient $C_1 = 1$. In f^0, the defender's strategy is taking edges $E_{1,1,1}$ and $E_{2,2,1}$ with probability 0.5 and the attacker's maximum expected utility is U_0, which can be achieved around time $(t_1 + t_2)/2$ when neither of the two edges $E_{1,1,1}$ and $E_{2,2,1}$ are within the target's protection range. If we adjust the flows to edge $E_{1,2,1}$ and $E_{2,1,1}$, as shown in Figure 8(b), the attacker's maximum expected utility in $[t_1, t_2]$ is reduced to $0.5U_0$ as edge $E_{1,2,1}$ is within the target's protection range all the way. So a rational attacker who is constrained to attack between $[t_1, t_2]$ will get a lower expected utility given defender strategy f^1 than given f^0, and thus the equilibrium with f^1 is more robust to this kind of deviation on the attacker side.



(a) f^0: the patroller is taking edges $E_{1,1,1}$ and $E_{2,2,1}$ with probability 0.5.

(b) f^1: the patroller is taking edges $E_{1,2,1}$ and $E_{2,1,1}$ with probability 0.5.

Figure 8: An example of flow adjust. An rational attacker who is constrained to attack in $[t_1, t_2]$ will choose to attack around time $(t_1 + t_2)/2$ to get utility U_0 given f^0 and attack around t_1 or t_2 to get utility $0.5U_0$ given f^1.

So in flow-adjust, we construct $M - 1$ new linear programs, one for each time interval $[t_{k^*}, t_{k^*+1}]$, $k^* = 1 \dots M - 1$ to find a new set of flow distribution probabilities $f(i, j, k^*)$ to achieve the lowest local maximum in $[t_{k^*}, t_{k^*+1}]$ with unchanged $p(i, k^*)$ and $p(i, k^* + 1)$.

The linear program for an interval $[t_k^*, t_{k^*+1}]$ is shown below.

$$
\begin{aligned}
&\min_{f(i,j,k^*)} v \\
&f(i,j,k^*) = 0, \text{ if } |d_j - d_i| > v_m * \delta t \\
&p(i,k^*+1) = \sum_{j=1}^{n} f(j,i,k^*), \forall i \in \{1 \dots n\} \\
&p(i,k^*) = \sum_{j=1}^{n} f(i,j,k^*), \forall i \in \{1 \dots n\} \\
&v \geq AttEU(F_q, t_k), \forall q \in \{1 \dots L\}, k \in \{k^*, k^*+1\} \\
&v \geq \max\{AttEU(F_q, \theta_{qk^*}^{r+}), AttEU(F_q, \theta_{qk^*}^{(r+1)-})\} \\
&\qquad \forall q \in \{1 \dots L\}, r \in \{0 \dots M_{qk^*}\}
\end{aligned}
$$

While the above linear program appears similar to the linear program of CASS, they have significant differences. Unlike CASS, the marginal probabilities $p(i,k^*)$ here are known constants and are provided as input and as mentioned above, there is a separate program for each $[t_{k^*}, t_{k^*+1}]$. Thus, we get $f(i,j,k^*)$ such that the local maximum in $[t_{k^*}, t_{k^*+1}]$ is minimized. Denote the minimum as $v_{k^*}^1$. From the original flow distribution f^0, we get $AttEU_{f^0}(F_q, t)$ and we denote the original local maximum value in $[t_{k^*}, t_{k^*+1}]$ as $v_{k^*}^0$. As the subset $\{f^0(i,j,k^*)\}$ of the original flow distribution f^0 is a feasible solution of the linear program above, we have $v_{k^*}^1 \leq v_{k^*}^0$, noting that the equality happens for the interval from which the attacker's best response is chosen.

Note that any change made to $f(i,j,k)$ in an interval $[t_k^*, t_{k^*+1}]$ will not affect the performance of f in other intervals as the marginal probabilities $p(i,k)$ are kept the same, i.e., changing $f(i,j,k^*)$ based on the linear program above is independent from any change to $f(i,j,k), k \neq k^*$. So we can solve the $M-1$ linear programs independently. After calculating $f(i,j,k^*)$ for all $k^* = 1..M-1$, we can get the new defender strategy f^1 by combining the solutions $f(i,j,k^*)$ of the different linear programs together. As $v_{k^*}^1 \leq v_{k^*}^0$, we have

$$
\max_{q \in \{1 \dots L\}, t \in [t_{k^*}, t_{k^*+1}]} AttEU_{f^0}(F_q, t) \leq \max_{q \in \{1 \dots L\}, t \in [t_{k^*}, t_{k^*+1}]} AttEU_{f^1}(F_q, t)
$$

for all $k^* = 1..M-1$, i.e., f^1 locally dominates f^0.

On the other hand, while we have restricted the strategies to have the same $p(i,k)$, there may exist another strategy f^2 with a different set of $p(i,k)$ that locally dominates f^1. Finding locally dominating strategies with different $p(i,k)$ from the original is a topic of future research.

Although the two refinement approaches we provide do not necessarily lead to a non-dominated strategy under the corresponding dominance definition, these two approaches are guaranteed to find a more robust (or at least indifferent) equilibrium when faced with constrained attackers compared to the original equilibrium we obtain from CASS. Clearly, these two refinement approaches do not exhaust the space of refinement approaches — other refinement approaches are possible that may lead to other equilibria that are better than (e.g. dominate) the one found by CASS. However, it is likely that different defender strategies resulting from different equilibrium refinements are not comparable to each other in terms of dominance, i.e., with some constrained attackers, one equilibrium might turn out to be better and with other constrained attackers, another equilibrium might be better. Their computational costs may differ as well. Thus, understanding this space of refinement approaches in terms of their computational cost and output quality, and determining which approach should be adopted under which circumstances is an important challenge for future work.

5. Extension To Two-Dimensional Space

Both DASS and CASS presented in Section 3 are based on the assumption that both the targets and the patrollers move along a straight line. However, a more complex model is needed in some practical domains. For example, Figure 9 shows a part of the route map of Washington State Ferries, where there are several ferry trajectories. If a number of patroller boats are tasked to protect all the ferries in this area, it is not necessarily optimal to simply assign a ferry trajectory to each of the patroller boat and calculate the patrolling strategies separately according to CASS described in Section 3. As the ferry trajectories are close to each other, a patrolling strategy that can take into account all the ferries in this area will be much more efficient, e.g., a patroller can protect a ferry moving from Seattle to Bremerton first, and then change direction halfway and protect another ferry moving from Bainbridge Island back to Seattle.

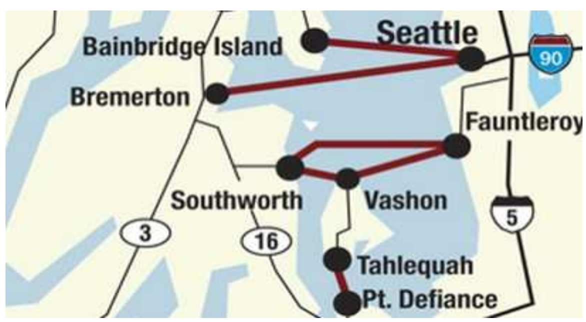


Figure 9: Part of route map of Washington State Ferries

In this section, we extend the previous model to a more complex case, where the targets and patrollers move in a two-dimensional space and provide the corresponding linear-program-based solution. Again we use a single defender resource as an example, and generalize to multiple defenders at the end of this section.

5.1 Defender Strategy for 2-D

As in the one-dimensional case, we need to discretize the time and space for the defender to calculate the defender's optimal strategy. The time interval T is discretized into a set of time points $T = \{t_k\}$. Let $G = (V, E)$ represents the graph where the set of vertices V corresponds to the locations that the patrollers may be at, at the discretized time points in T, and E is the set of feasible edges that the patrollers can take. An edge $e \in E$ satisfies the maximum speed limit of patroller and possibly other practical constraints (e.g., a small island may block some edges).

5.2 DASS for 2-D

When the attack only occurs at the discretized time points, the linear program of DASS and described in Section 3 can be applied to the two-dimensional settings when the distance in Constraint 9 is substituted with Euclidean distance in 2-D space of nodes V_i and V_j.

$$\min_{f(i,j,k),p(i,k)} \quad v \tag{30}$$

$$f(i,j,k) \in [0,1], \forall i,j,k \tag{31}$$

$$f(i,j,k) = 0, \forall i,j,k \text{ such that } ||V_j - V_i|| > v_m \delta t \tag{32}$$

$$p(i,k) = \sum_{j=1}^{N} f(j,i,k-1), \forall i, \forall k > 1 \tag{33}$$

$$p(i,k) = \sum_{j=1}^{N} f(i,j,k), \forall i, \forall k < M \tag{34}$$

$$\sum_{i=1}^{N} p(i,k) = 1, \forall k \tag{35}$$

$$v \geq \text{AttEU}(F_q, t_k), \forall q, \forall k \tag{36}$$

Note that $f(i,j,k)$ now represents the probability that a patroller is moving from node V_i to V_j during $[t_k, t_{k+1}]$. Recall in Figure 2.1, a patroller protects all targets within her protective circle of radius r_e. However, in the one-dimensional space, we only care about the straight line AB, so we used $\beta_q(t) = [\max\{S_q(t) - r_e, d_1\}, \min\{S_q(t) + r_e, d_N\}]$ as the protection range of target F_q at time t, which is in essence a line segment. In contrast, here the whole circle needs to be considered as the protection range in the two-dimensional space and the extended protection range can be written as $\beta_q(t) = \{V = (x,y) : ||V - S_q(t)|| \leq r_e\}$. This change affects the value of $I(i,q,k)$ and thus the value of $AttEU(F_q, t_k)$ in Constraint 36.

5.3 CASS for 2-D

When the attacking time t can be chosen from the continuous time interval T, we need to analyze the problem in a similar way as in Section 3.3. The protection radius is r_e, which means only patrollers located within the circle whose origin is $S_q(t)$ and radius is r_e can protect target F_q. As we assume that the target will not change its speed and direction during time $[t_k, t_{k+1}]$, the circle will also move along a line in the 2-D space. If we track the circle in a 3-D space where the x and y axes indicate the position in 2-D and the z axis is the time, we get an oblique cylinder, which is similar to a cylinder except that the top and bottom surfaces are displaced from each other (See Figure 10). When a patroller moves from vertex $V_i (\in V)$ to vertex V_j during time $[t_k, t_{k+1}]$, she protects the target only when she is within the surface. In the 3-D space we described above, the patroller's movement can be represented as a straight line.

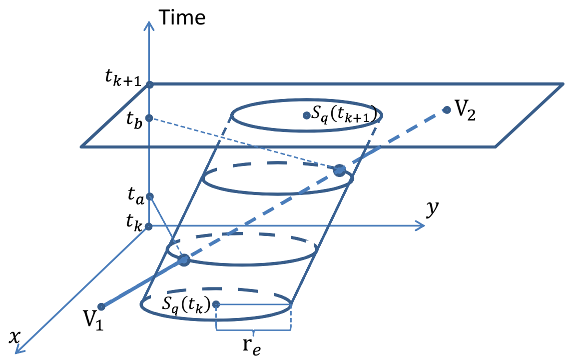


Figure 10: An illustration of the calculation of intersection points in the two-dimensional setting. The x and y axes indicates the position in 2-D and the z axis is the time. To simplify the illustration, z axis starts from time t_k. In this example, there are two intersection points occurring at time points t_a and t_b.

Intuitively, there will be at most two intersection points between the patroller's route in 3-D space and the surface. This can be proved by analytically calculating the exact time of these intersection points. Assume the patroller is moving from $V_1 = (x_1, y_1)$ to $V_2 = (x_2, y_2)$ and the target is moving from $S_q(t_k) = (\hat{x_1}, \hat{y_1})$ to $S_q(t_{k+1}) = (\hat{x_2}, \hat{y_2})$ during $[t_k, t_{k+1}]$ (an illustration is shown in Figure 10). To get the time of the intersection points, we solve a quadratic equation with these coordination parameters and protection radius r_e. We present the detailed calculation in Appendix B. If a root of the quadratic equation is within the interval $[t_k, t_{k+1}]$, it indicates that the patroller's route intersects with the surface at this time point. So there will be at most two intersection points. Once we find all these intersection points, the same analysis in Section 3.3 applies and we can again claim Lemma 1. So we conclude that we only need to consider the attacker's strategies at these intersection points. We use the same notation θ^r_{qk} as in the one-dimensional case to denote the sorted intersection points and get the following linear program for the 2-D case.

$$\min_{f(i,j,k),p(i,k)} \quad v \tag{37}$$

subject to constraints(31 . . . 36)

$$v \geq \max\{\text{AttEU}(F_q, \theta^{r+}_{qk}), \text{AttEU}(F_q, \theta^{(r+1)-}_{qk})\} \tag{38}$$
$$\forall k \in \{1 \ldots M\}, q \in \{1 \ldots L\}, r \in \{0 \ldots M_{qk}\}$$

Algorithm 1 can still be used to add constraints to the linear program of CASS for the 2-D case. The main difference compared to CASS in the 1-D case is that since Euclidean distance in 2-D is used in Constraint 32 we need to use the extended definition of $\beta_q(t)$ in 2-D when deciding the entries in the coefficient matrix $A^r_{qk}(i, j)$.

For multiple defender resources, again the linear program described in Section 3.4 is applicable when the extended definition of $\beta_q(t)$ is used to calculate $AttEU$ and Constraint 19 is substituted with the following constraint:

$$f(i_1, j_1, \ldots, i_W, j_W, k) = 0, \forall i_1, \ldots, i_W, j_1, \ldots, j_W \text{ such that } \exists u, \|V_{j_u} - V_{i_u}\| > v_m \delta t.$$

6. Route Sampling

We have discussed how to generate an optimal defender strategy in the compact representation; however, the defender strategy will be executed as taking a complete route. So we need to sample a complete route from the compact representation. In this section, we give two methods of sampling and show the corresponding defender strategy in the full representation when these methods are applied.

The first method is to convert the strategy in the compact representation into a Markov strategy. A Markov strategy in our setting is a defender strategy such that the patroller's movement from t_k to t_{k+1} depends only on the location of the patroller at t_k. We denote by $\alpha(i, j, k)$ the conditional probability of moving from d_i to d_j during time t_k to t_{k+1} given that the patroller is located at d_i at time t_k. In other words $\alpha(i, j, k)$ represents the chance of taking edge $E_{i,j,k}$ given that the patroller is already located at node (t_k, d_i). Thus, given a compact defender strategy specified by $f(i, j, k)$ and $p(i, k)$, we have

$$\alpha(i, j, k) = f(i, j, k)/p(i, k), \text{ if } p(i, k) > 0. \tag{39}$$

$\alpha(i, j, k)$ can be an arbitrary number if $p(i, k) = 0$. We can get a sampled route by first determining where to start patrolling according to $p(i, 1)$; then for each t_k, randomly choose where to go from t_k to t_{k+1} according to the conditional probability distribution $\alpha(i, j, k)$. The distribution from this sampling procedure matches the given marginal variables as each edge $E_{i,j,k}$ is sampled with probability $p(i, k)\alpha(i, j, k) = f(i, j, k)$. This sampling method actually leads to a full representation where route $R_u = (d_{r_u(1)}, d_{r_u(2)}, ..., d_{r_u(M)})$ is sampled with probability $p(r_u(1), 1) \prod_{k=1}^{M-1} \alpha(r_u(k), r_u(k + 1), k)$, the product of the probability of the initial distribution and the probability of taking each step. This method is intuitively straightforward and the patrol route can be decided online during the patrol, i.e., the position of the patroller at t_{k+1} is decided when the patroller reaches its position at t_k, which makes the defender strategy more unpredictable. The downside of the method is that the number of routes chosen with non-zero probability can be as high as N^M. For 2-D case, the patroller is located at node V_i at time t_k. The sampling process is exactly the same when $\alpha(i, j, k)$ is used to denote the probability of moving from V_i to V_j during $[t_k, t_{k+1}]$.

The second method of sampling is based on the decomposition process mentioned in Section 4.1 (step (i)). As we discussed above for the first sampling method, sampling is essentially restoring a full representation from the compact representation. As shown in Table 1, there are multiple ways to assign probabilities to different routes and the decomposition process of "route-adjust" constructively defines one of them. So we can make use of the information we get from the process, and sample a route according to the probability assigned to each decomposed route. The number of routes chosen with non-zero probability is at most N^2M, much less than the first method and thus it becomes feasible to describe the strategy in full representation, by only providing the routes that are chosen with positive probability. Different sampling approaches may be necessitated by different application requirements. Some applications might require that the defender obtain a strategy in full representation and only be presented a small number of pure strategies. However, for other applications, a strategy that can be decided on-line, potentially with a hand-held smartphone such as in (Luber, Yin, Fave, Jiang, Tambe, & Sullivan, 2013) may be preferred. Therefore, based on the needs of the application, different sampling strategies might be selected.

7. Evaluation

We use different settings in the ferry protection domain and compare the performance in terms of the attacker's expected utility $\text{AttEU}(F_q, t)$. As it is a zero-sum game, a lower value of AttEU indicates a higher value of defender's expected utility.

We will run experiments both for 1-D and 2-D setting. We will evaluate the performance of CASS and show the sampling results. We will also evaluate the improvement of the two refinement approaches for 1-D. Section 7.1 shows our results for the 1-D setting; Section 7.2 for the 2-D setting.

7.1 Experiments for One Dimensional Setting

For 1-D setting, we first evaluate the performance of the solvers and then show how much the performance can be improved by using the refinement methods. We also show sampled routes for an example setting and evaluate CASS for varying number of patrollers.

7.1.1 Experimental Settings

We used the following setting for the experiments in one dimensional case. This is a complex spatio-temporal game; rather than a discrete security game as in most previous work. There are three ferries moving between terminals A and B and the total distance $AB = 1$. The simulation time is 30 *minutes*. The schedules of the ferries are shown in Figure 11, where the x-axis indicates the time and the y-axis is the distance from terminal A. Ferry 1 and Ferry 3 are moving from A to B while Ferry 2 is moving from B to A. The maximum speed for patrollers is $v_m = 0.1/min$ and the protection radius is $r_e = 0.1$. Experiments in the one-dimensional case are using 2 patrollers (where $C_1 = 0.8$, and $C_2 = 1.0$), except in Section 7.1.5 where we report on experiments with different numbers of patrollers.

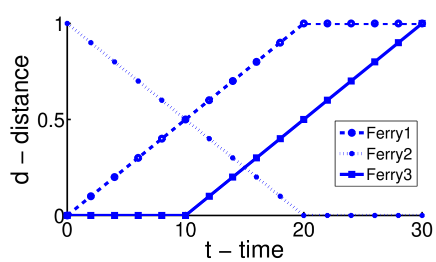


Figure 11: Schedules of the ferries

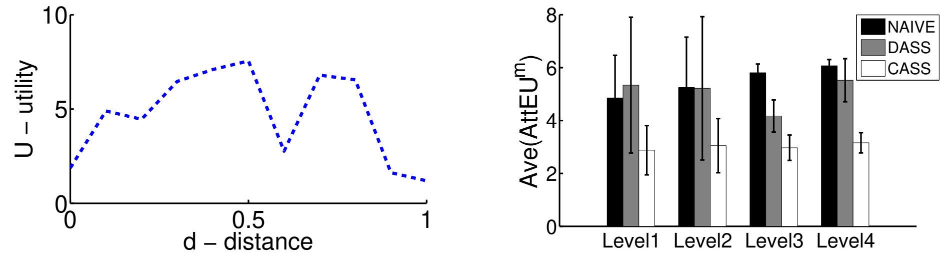
7.1.2 Performance of Solvers

We compare the strategies calculated by CASS with DASS and a baseline strategy. In the baseline strategy, the two patrollers choose a ferry with a probability of 1/3 (uniformly random) and move alongside it to offer it full protection, leaving the other two unprotected (strategy observed in practice). First we wished to stress-test CASS by using more complex utility functions than in the realistic case that follows. Therefore, we tested under 4 different discretization levels (details about discretization levels are included in Table 4) with random utilities, and at each discretization level, we created 20 problem instances. The problem instances are different across levels. In this ferry protection domain, the utility function for each ferry usually depends on the ferry's position, so each instance has utilities uniformly randomly chosen between $[0, 10]$ at discretized distance points; an example is shown in Figure 12(a). The chosen discretization levels have ensured that $U_q(t)$ is linear in t in each time interval $[t_k, t_{k+1}]$ for each target F_q. In Figure 12(a), the x-axis indicates the distance d from terminal A, the y-axis indicates the utility of a successful attack if the ferry is located at distance d. In Figure 12(b), x-axis plots the four discretization levels and y-axis plots the average attacker expected utility if he plays best response over the 20 instances for baseline, DASS and CASS. CASS is shown to outperform DASS and baseline and the differences are statistically significant ($p < 0.01$). Note that different sets of instances are generated for different discretization levels, so we cannot compare the results across levels directly. However, it is helpful in better understanding the models. From the figure, we find the solution quality of DASS varies a lot and sometimes can be worse than the naive strategy (e.g., level 1). This is because DASS calculates an optimal solution that considers only the attacks at the discretized time points. In Figure 12(b), the solution quality is measured by $AttEU^m$, which is calculated as the maximum over the continuous attacker strategy set. The gap between the optimal objective function of DASS and the actual $AttEU^m$ given the optimal solution of DASS may vary for different strategies and different discretization levels. Another interesting observation is that the average solution quality of CASS is almost the same for all discretization levels. Despite the difference in instance sets, this result implies that the improvement of a finer discretization may be limited for CASS.

Level	δt (minutes)	M	δd	N
1	10	4	0.5	3
2	5	7	0.25	5
3	2.5	13	0.125	9
4	2	16	0.1	11

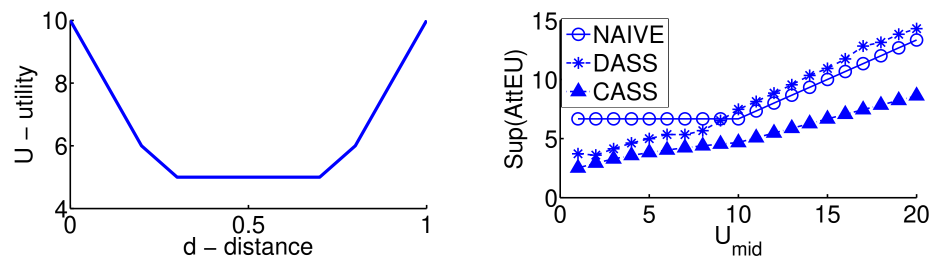
Table 4: Details about discretization levels. In the experiments mentioned in this section, the distance space is evenly discretized, parameterized by $\delta d = d_{i+1} - d_i$.

Next we turn to *more realistic* utility function in this ferry domain, which is of U-shape or inverse U-shape. Figure 13(a) shows a sample utility curve where the attacker gains higher utility closer to the shore. We fix the utility at the shore as 10, vary the utility in the middle (denoted as U_{mid}), which is the value on the floor of the U-shape or the top of the inverse U-shape and evaluate the strategies. In Figure 13(b), U_{mid} is shown on x-axis



(a) Randomized attacker utility function (b) Average solution quality of different strategies

Figure 12: Performance under different randomized utility function settings. The utility function in this set of experiments is a function of the distance to Terminal A. The utility function is piece-wise linear and the value at discretized distance points d_i is chosen randomly between [0,10].



(a) Realistic attacker utility function with $U_{mid} = 5$ (b) Solution quality of different strategies

Figure 13: Performance under different realistic utility function settings. The utility function is U-shape or inverse U-shape. The utility around distance 0.5 is denoted as U_{mid}. We compare the defender strategy given by DASS and CASS with the baseline when U_{mid} is changing from 1 to 20.

and we compare performance of the strategies in terms of attacker's expected utility when he plays best response on the y-axis. We conclude that 1) the strategy calculated by CASS outperforms the baseline and DASS; 2) DASS may actually achieve worse results than the baseline.

Among all these different experiment settings of discretization and utility function, we choose one instance and provide a more detailed analysis for it. We refer to this instance as *example setting* in the following of this section. In this example setting, discretization level 4 is used and the utility curve is as shown in Figure 13(a), other parameters involved are described in Section 7.1.1. Figure 14 compares the attacker expected utility function when DASS and CASS is used respectively. The x-axis indicates the time t, and the y-axis indicates the attacker's expected utility if he attacks Ferry 1 at time t. For the strategy calculated by DASS, the worst performance at discretized time points is 3.50 ($\mathrm{AttEU}(F_1, 20)$), however, the supremum of $\mathrm{AttEU}(F_1, t)$, $t \in [0, 30]$ can be as high as 4.99 ($\mathrm{AttEU}(F_1, 4^+)$),

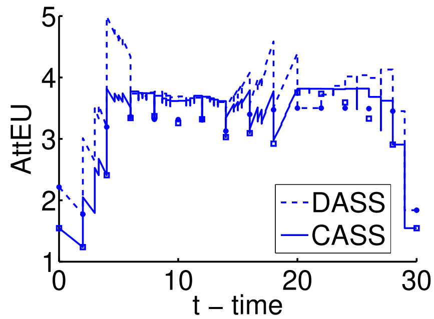


Figure 14: The attacker's expected utility function given the defender strategy calculated by DASS vs CASS under example setting. The expected utilities at the discretized time points are indicated by squares for CASS and dots for DASS. The maximum of *AttEU* under CASS is 3.82, 30%less than the maximum of *AttEU* under DASS, which is 4.99.

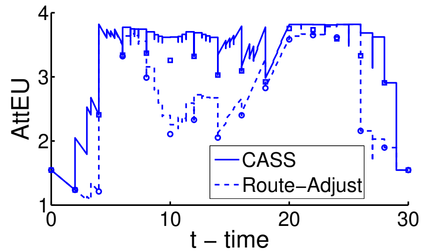
which experimentally shows that taking into consideration the attacks between the discretized time points is necessary. For the strategy calculated by CASS, the supremum of AttEU(F_1, t) is reduced to 3.82.

7.1.3 Improvement Using Refinement Methods

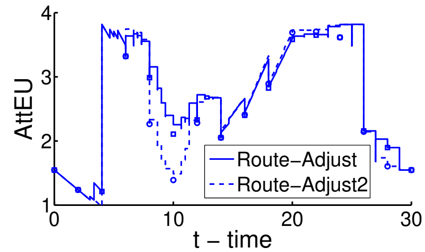
We compare the refinement approaches described in Section 4 and analyze the tradeoff between performance improvement and runtime. Three approaches are considered for comparison: route-adjust, flow-adjust and a variation of route-adjust, denoted by route-adjust2. In step (ii) of route-adjust, we replace every node in the route one-by-one in sequence.[3] In step (ii) of route-adjust2, we replace every consecutive pair of nodes in the route in sequence.

We first show results for the example setting. In Figure 15(a), we compare the AttEU(F_q, t) function of the defender strategy given by CASS and of the one after route-adjust for Ferry 1. It shows for an attack aiming at any target at any time, the defender strategy after route-adjust refinement is equally good or better than the one in the original equilibrium, and thus the defender performs equally or better no matter how the attacker is constrained in time, i.e., the defender strategy after route-adjust dominates the original strategy. Figure 15(b) is the comparison between AttEU function of the defender strategy after route-adjust and the one after route-adjust2 for Ferry 1. The one after route-adjust2 does not dominate the one after route-adjust but overall the former appears to perform better than the latter more frequently and by larger amounts. If we use the average value of AttEU function as a metric of performance, we will show that route-adjust2 is better than route-adjust in this example setting later in Table 5. Figure 15(c) shows the comparison between the AttEU function of the defender strategy given by CASS and that of the defender strategy after

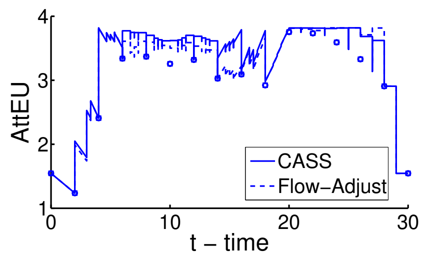
3. In supplementary experiments, we also tested route-adjust with more iterations, e.g., repeating the process of replacing every node in sequence five times. The extra benefit is insignificant while the runtime increases proportionally to the number of iterations. In light of this, we choose to replace each node only once in the experiments reported in this article.



(a) AttEU function of Ferry 1 after route-adjust (one node at a time)



(b) AttEU function of Ferry 1 after route-adjust2 (two nodes at a time)



(c) Performance of flow-adjust

Figure 15: Performance of equilibrium refinement approaches.

flow-adjust for Ferry 1. The strategy given by CASS is not dominated by the one after flow-adjust under Definition 7, but if we investigate the maximum of AttEU in each time interval $[t_k, t_{k+1}]$, as shown in Table 6, we find that the defender strategy after flow-adjust locally dominates the original strategy.

We list the worst case performance and the average performance of AttEU function over all ferries in this example setting for four defender strategies (CASS, route-adjust, route-adjust2, flow-adjust) in Table 5, from which we conclude that 1) the worst case performance of all strategies of flow-adjust is the same, which means the defender achieves exactly same expected utility towards an unconstrained rational attacker; 2) the average performance of flow-adjust is slightly better than the CASS, but is outperformed by route-adjust and route-adjust2, while it takes much less time to run compared to the other two; 3) in this example setting, when we adjust two consecutive nodes at a time, the performance is better than adjusting only one node at a time, but the difference is not significant and it is much more expensive in terms of run-time.

Strategies	Worst Case Performance	Average Performance	Runtime (minutes)
CASS	3.82	3.40	-
Route-Adjust	3.82	2.88	8.96
Route-Adjust2	3.82	2.76	32.31
Flow-Adjust	3.82	3.34	0.50

Table 5: Comparison of different refinement approaches in terms of average performance and runtime. Only the runtime for the refinement process is calculated.

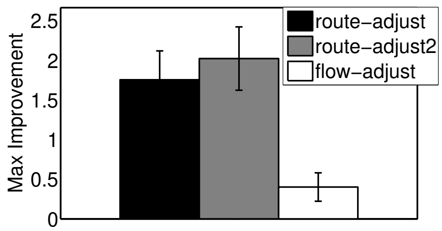
time interval $[t_k, t_{k+1}]$	maximum before	maximum after	time interval $[t_k, t_{k+1}]$	maximum before	maximum after
$[2, 4]$	3.7587	3.6675	$[16, 18]$	3.8111	3.7291
$[4, 6]$	3.8182	3.8182	$[18, 20]$	3.8182	3.8182
$[6, 8]$	3.8153	3.6164	$[20, 22]$	3.8182	3.8182
$[8, 10]$	3.8137	3.6316	$[22, 24]$	3.8182	3.8182
$[10, 12]$	3.8052	3.6316	$[24, 26]$	3.8182	3.8182
$[12, 14]$	3.8050	3.5664	$[26, 28]$	3.8182	3.8182
$[14, 16]$	3.7800	3.2100	$[28, 30]$	3.8182	3.8182

Table 6: The maximum of attacker's expected utility in each time interval decreases after flow-adjust is used.

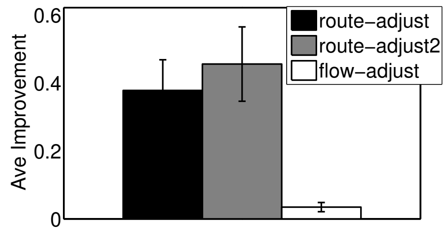
Figure 16(a) and Figure 16(b) shows the maximum and the average improvement of route-adjust, route-adjust2 and flow-adjust, averaged over all the 20 instances of Level 4 with randomized utilities that have been used for Figure 12(b); and Figure 16(c) shows the average runtime. The maximum improvement is the largest difference between the AttEU function given defender strategy calculated by CASS and the one after refinement. The average improvement is the average difference between the two functions. The standard deviations over all instances are shown as error bars. Figure 16 confirms that all the refinement approaches improve the defender strategy calculated by CASS in terms of both the maximum performance and average performance and thus provide better defender strategies given possible constrained attackers. Route-adjust2 achieves the most improvement, then route-adjust, and flow-adjust the least. Flow-adjust achieves much less improvement compared to the other two approaches. One explanation for this is that the constraints are very strong as they require all marginal probabilities to be unchanged so it is likely that little changes are made to the original defender strategy. The difference between route-adjust2 and route-adjust is not as significant. In terms of run-time, flow-adjust is the least expensive, route-adjust the second and route-adjust2 the most. Route-adjust2 is significantly more expensive compared to the other two. So we conclude that route-adjust is a better choice considering the tradeoff between improvement and the runtime.

7.1.4 Sampled Routes

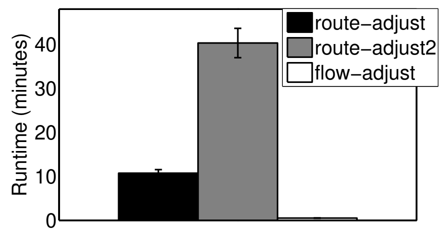
We first convert the defender strategy under the example setting into a Markov strategy and sample 1000 pair of patrol routes. The defender strategy used here is the one after "route-adjust". In each sample, a pair of routes is chosen step by step for the two patrol boats according to the joint conditional probability distribution $\{\alpha(i1, j1, i2, j2, k)\}$. The routes for the two patrol routes are chosen simultaneously as they are coordinating with each other. We cannot show each pair separately for all 1000 samples. Instead, Figure 17(a) shows the frequency of being taken out of the 1000 samples of each edge. The x-axis indicates the time and the y-axis is the distance to terminal A. The width of the each edge indicates the frequency of being chosen by at least one patroller. Although Figure 17(a) does not precisely depict the samples, it provides a rough view of how the routes are taken by the patrol boats.



(a) Average of maximal improvement



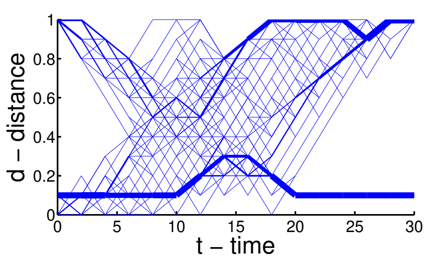
(b) Average of average improvement



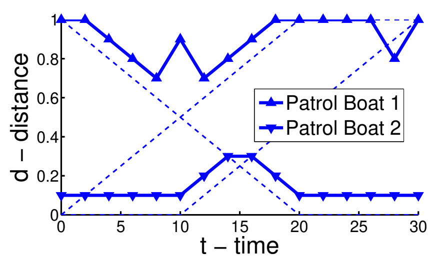
(c) Average of runtime

Figure 16: Comparison of refinement approaches.

Figure 17(b) shows the pair of routes that is of highest probability when we use the decomposition method of sampling. The solid lines show the patrol boats' routes and the dashed lines show the ferries' schedules. We get 3958 different pair of patrol routes in total in the decomposition process and the shown pair of routes is chosen with probability 1.57%.

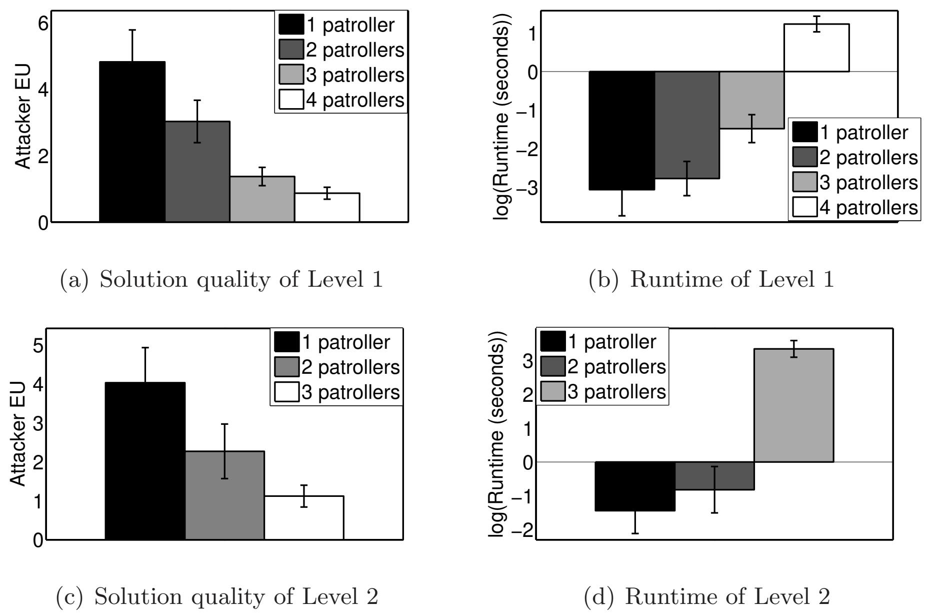


(a)



(b)

Figure 17: Results for sampling under the example setting: (a) Frequency of each edge is chosen when the first sampling method based on Markov strategy is used. (b) Decomposed routes with highest probability superimposed on ferry schedules when the second sampling method based on decomposition is used.



(a) Solution quality of Level 1

(b) Runtime of Level 1

(c) Solution quality of Level 2

(d) Runtime of Level 2

Figure 18: Performance with varying number of patrollers.

7.1.5 Number of Patrollers

Figure 18(a) shows the improvement in performance of CASS with increasing number of patrollers under discretization Level 1. The x-axis shows the number of patrollers and the y-axis indicates the average of attacker's maximal expected utility, i.e., the expected reward when he plays his best response. The results are averaged over 20 random utility settings of discretization Level 1. With fewer patrollers, the performance of the defender varies a lot depending on the randomized utility function (as indicated by standard deviation shown as the error bar). But the variance gets much smaller with more patrollers, which means the defender has sufficient resources for different instances. Figure 18(b) shows the run-time for CASS. The y-axis indicates the average of natural logarithm of runtime. Not surprisingly, the run-time increases when the number of patrollers increases.

Figure 18(c) and 18(d) show the average performance and run-time of CASS with discretization Level 2, using the same set of utility settings as used in Level 1. Only results for 1 to 3 patrollers are shown. The program runs out of memory for 4 patrollers as there are $N^8M = 2734375$ flow distribution variables and at least $N^4M = 8757$ constraints. Note that the average solution quality of Level 2 is better than the result of Level 1 (e.g., the average attacker EU for 1 patroller is 4.81 in Level 1 and 4.13 in Level 2), which indicates a higher level of granularity can improve the solution quality. However, granularity clearly affect the ability to scale-up; which means that we need to consider the tradeoff between the solution quality and the memory used and one way to combat the scaling-up problem is to reduce the level of granularity. Nonetheless, the number of patrollers we have encountered in real-world scenarios such as at New York is of the order of 3 or 4, so CASS is capable at least for key real-world scenarios.

7.1.6 Approximation Approach for Multiple Defender Resources

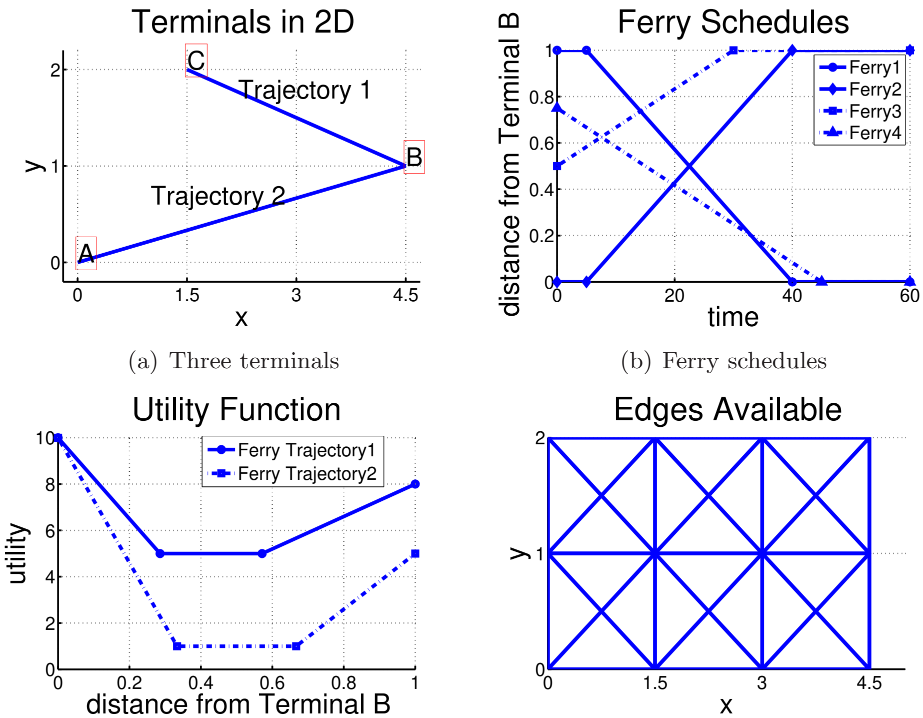
We tested the first approximation approach for multiple defender resources described in Section 3.4 for the example setting. We used the **fmincon** function with interior-point method in MATLAB to minimize the non-linear objective function (Equation 25). Table 7 lists different run-time and the value of the objective function achieved given different iteration number (denoted as *MaxIter*). The function is not ensured to provide a feasible solution when the iteration number is not large enough, as shown in the first two rows. We compared the result with our LP formulation of DASS, which was implemented in MATLAB using **linprog** function. DASS can be solved within 8.032 seconds and provides an optimal solution $\text{AttEU}^m = 3.5$, this approximation approach is outperformed in both run-time efficiency and solution quality. This approach fails to provide a feasible solution efficiently and even when sufficient time is given (more than 400 times the run-time of the LP formulation), the maximum attacker expected utility is 18% larger than the optimal solution. This is mainly because the new formulation in the approximation approach is no longer linear or convex, making it difficult to find a global maximum.

$MaxIter$	$Run-time(sec)$	AttEU^m
3000	4.14	infeasible
10000	17.21	infeasible
900000	3298	4.0537

Table 7: Performance of approximation approach.

7.2 Experiments for Two Dimensional Setting

The settings in 2-D space are more complex even with single patroller. Here we show an example setting motivated by the ferry system between Seattle, Bainbridge island and Bremerton as shown in Figure 9. In this example setting, three terminals (denoted as A,B and C) are non-collinear in the 2-D space as shown in the Figure 19(a). Ferry 1 and Ferry 2 are moving on the trajectory between Terminal B and C (denoted as Trajectory 1) and Ferry 3 and Ferry 4 are moving on the trajectory between Terminal B and A (denoted as Trajectory 2). The schedules of the four ferries are shown in Figure 19(b), where the x-axis is the time and the y-axis is the distance from the common terminal B. Ferry 1 moves from C to B, Ferry 2 moves from B to C, Ferry 3 moves from B to A and Ferry 4 moves from A to B. Similar to the one-dimensional scenario in ferry domain, we assume the utility is decided by the ferry's position and the utility function is shown in Figure 19(c). The x-axis is the distance from the common terminal B and the y-axis is the utility for the two trajectories respectively. The 2-D space is discretized into a grid as shown in Figure 19(d) with $\delta x = 1.5$ and $\delta y = 1$ indicating the interval in the x-axis and y-axis. A patroller will be located at one of the intersection points of the grid graph at any discretized time points. The simulation time is 60 minutes and $M = 13$, i.e., $t_{k+1} - t_k = 5$ minutes. The speed limit for the patroller is $v_e = 0.38$ and all the available edges that a patroller can take during $[t_k, t_{k+1}]$ are shown in Figure 19(d). Only one patroller is involved. The protection radius is set to $r_e = 0.5$, and protection coefficient is $C_1 = 0.8$.



(a) Three terminals

(b) Ferry schedules

(c) Utility function

(d) Available edges

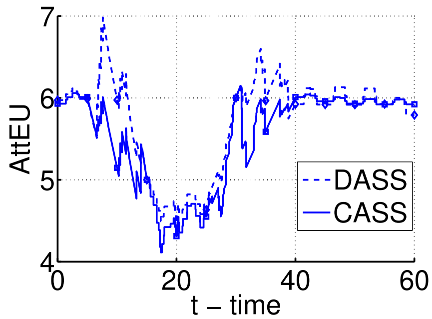
Figure 19: An example setting in two-dimensional space

Figure 20(a) compares the performance of DASS and CASS for Ferry 2. Ferry 2 is chosen because in both strategies, the attacker's best response is to attack Ferry 2. The x-axis is the time t, and the y-axis is the attacker expected utility of attacking Ferry 1 at time t. The maximum of AttEU of CASS is 6.1466, 12% lower compared to the result of DASS, which is 6.9817. Figure 20(b) and 20(c) show two sampled route given the strategy calculated by CASS on the 2-D map where the dashed lines represents for the ferry trajectories. The patroller starts from the node with text "start" and follows the arrowed route, and ends at the node with text "end" at the end of the patrol. She may stay at the nodes with text "stay". The patrol routes are shown in a intuitive way but can be ambiguous. The exact route should be listed as a table with time and position. The routes are sampled based on the converted Markov strategy, and the total number of patrol routes that may be chosen with non-zero probability is 4.49×10^{10}.

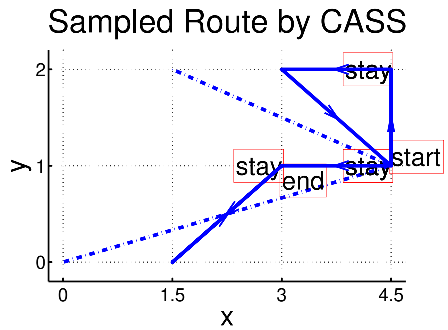
8. Related Work

In this section we discuss literature related to our work. We will first discuss work on the computation of game-theoretic patrolling strategies, then discuss work on continuous games, and finally discuss work on equilibrium refinement.

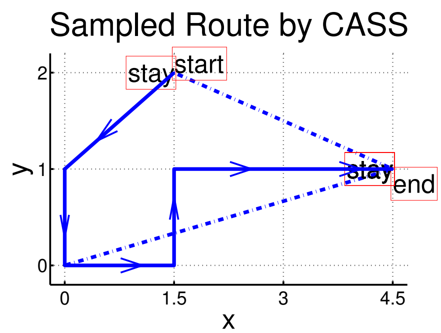
As mentioned in the introduction, Stackelberg games have been widely applied to security domains, although most of this work has considered static targets (e.g., Korzhyk et al., 2010; Krause, Roper, & Golovin, 2011; Letchford & Vorobeychik, 2012; Kiekintveld



(a) Solution quality of DASSand CASS for Ferry 2



(b) Sampled route 1 superimposed on ferry trajectories



(c) Sampled route 2 superimposed on ferry trajectories

Figure 20: Experimental results under two-dimensional settings

et al., 2013). Agmon, Kraus, and Kaminka (2008) proposed algorithms for computing mixed strategies for setting up a perimeter patrol in adversarial settings with mobile robot patrollers. Similarly, Basilico, Gatti, and Amigoni (2009) computed randomized leader s-trategies for robotic patrolling in environments with arbitrary topologies. Even when both of the players are mobile, e.g., in hider-seeker games (Halvorson, Conitzer, & Parr, 2009), infiltration games (Alpern, 1992) or search games (Gal, 1980), the targets (if any) were assumed to be static. Tsai et al. (2009) applied Stackelberg games to the domain of scheduling federal air marshals on board flights. The targets (i.e., flights) in this domain are mobile, but the players are restricted to move along the targets to protect or attack them. This stationary nature leads to discrete game models with finite numbers of pure strategies.

Bošanský, Lisý, Jakob, and Pěchouček (2011) and Vaněk, Jakob, Hrstka, and Pěchouček (2011) studied the problem of protecting moving targets. However, they both considered a model in which the defender, the attacker and targets have discretized movements on a directed graph. Such discretization of attacker strategy spaces can introduce suboptimality in the solutions, as we have shown with DASS. We, in our work, generalize the strate-gy space of the attacker to the continuous realm and compute optimal strategies even in such a setting. Furthermore, while we provide an efficient and scalable linear formulation, Bošanský et al. presented a formulation with non-linear constraints, which faced problems scaling up to larger games even with a single defender resource.

Yin et al. (2012) considered the domain of patrolling in public transit networks (such as the LA Metro subway train system) in order to catch fare evaders. Because the players ride along trains that follow a fixed schedule, the domain is inherently discrete and they modeled the patrolling problem as a finite zero-sum Bayesian game. Yin et al. proposed a compact representation for defender mixed strategies as flows in a network. We adapt this compact representation idea to a continuous domain. In particular, in our domain we need to model the interaction between the defender's flow and attacker's continuous strategy space. Our proposed sub-interval analysis used spatio-temporal reasoning to efficiently reduce the problem into a finite LP.

Games with continuous strategy spaces have been well-studied in game theory. Much of the economics literature has focused on games whose equilibria can be solved analytically (and thus the question of computation does not arise), for example the classical theory of auctions (see e.g., Krishna, 2009). Recent computational approaches for the analysis and design of auctions have focused on discretized versions of the auction games (e.g., Thompson & Leyton-Brown, 2009; Daskalakis & Weinberg, 2012). There has been research on efficiently solving two-player continuous games with specific types of utility functions, such as zero-sum games with convex-concave utility functions (Owen, 1995) and separable continuous games with polynomial utility functions (Stein, Ozdaglar, & Parrilo, 2008). Johnson, Fang, and Tambe (2012) studied a continuous game model for protecting forests from illegal logging. In their model the target (i.e., the forest) is stationary, and with further simplifying assumptions (e.g., the forest having a circular shape) they were able to solve the game efficiently. In contrast to existing work, our game model has moving targets in a continuous domain, and the resulting utility functions are discontinuous and thus existing approaches are not applicable. Our CASS algorithm solves the game optimally without needing to discretize the attacker's strategy space.

There is an extensive literature on equilibrium refinement; however most existing work on the computation of equilibrium refinement focuses on finite games. For simultaneous-move finite games, solution concepts such as perfect equilibrium and proper equilibrium were proposed as refinements of Nash equilibrium (Fudenberg & Tirole, 1991). Miltersen and Sørensen (2007) proposed an efficient algorithm for computing proper equilibria in finite zero-sum games. For finite security games, An et al. (2011) proposed a refinement of Stackelberg equilibrium and techniques for computing such refinements. The resulting defender strategy is robust against possibilities of constrained capabilities of the attacker. These existing approaches rely on the finiteness of action sets, and is thus not applicable to our setting. Simon and Stinchcombe (1995) proposed definitions of perfect equilibrium and proper equilibrium for infinite games with continuous strategy sets, however they did not propose any computational procedure for the resulting solution concepts. Exact computation of equilibrium refinements of continuous games such as $\mathrm{MRMT_{sg}}$ remains a challenging open problem.

9. Conclusion

This paper makes several contributions in computing optimal strategies given moving targets and mobile patrollers. First, we introduce $\mathrm{MRMT_{sg}}$, a novel Stackelberg game model that takes into consideration spatial and temporal continuity. In this model, targets move with fixed schedules and the attacker chooses his attacking time from a continuous time interval. Multiple mobile defender resources protect the targets within their protection radius, and bring in continuous space in our analysis. Second, we develop a fast solution approach, CASS, based on compact representation and sub-interval analysis. Compact representations dramatically reduce the number of variables in designing the optimal patrol strategy for the defender. Sub-interval analysis reveals the piece-wise linearity in attacker expected utility function and shows there is a finite set of dominating strategies for the attacker. Third, we propose two approaches for equilibrium refinement for CASS's solutions: route-adjust and flow-adjust. Route-adjust decomposes the patrol routes, greedily improves the routes and composes the new routes together to get the new defender strategy. Flow-adjust is a fast and simple algorithm that adjusts the flow distribution to achieve optimality in each time interval while keeping the marginal probability at the discretized time points unchanged. Additionally, we provide detailed experimental analyses in the ferry protection domain. CASS has been deployed by the US Coast Guard since April 2013.

10. Future Work

There are several important avenues for future work. These include: (i) use a decreasing function to model the protection provided to the targets instead of using a fixed protection radius; (ii) handle practical constraints on patrol boat schedule as not all are easily implementable; (iii) efficiently handle more complex and uncertain target schedules and utility functions.

Here we provide an initial discussion about the relaxation of the assumptions that we listed in Section 2 and used throughout the paper:

- If we allow for complex and uncertain target schedules, we may model the problem as a game where the targets follow stochastic schedules. Our framework may still apply but may need to be enriched (e.g., using approaches such as use of MDPs to represent defender strategies, see Jiang, Yin, Zhang, Tambe, & Kraus, 2013). Coordinating multiple such defenders then becomes an important challenge. It may be helpful in such cases to appeal to more of the prior work on multi-agent teamwork, given the significant uncertainty in such cases leading to more need for on-line coordination (Tambe, 1997; Stone, Kaminka, Kraus, & Rosenschein, 2010; Kumar & Zilberstein, 2010; Yin & Tambe, 2011).

- If we focus on environments where multiple attackers can coordinate their attacks, then we may need to further enhance our framework. Prior results from Korzhyk, Conitzer, and Parr (2011) over stationary targets and discrete time would be helpful in addressing this challenge, although the case of moving targets in continuous space and time in such cases provides a very significant challenge. Combining with the previous item for future work, a complex multiple defender multiple attacker scenario would appear to be a very significant computational challenge.

Acknowledgments

We thank the USCG officers, and particularly Craig Baldwin, Joe Direnzo and Francis Varrichio and officers at sector New York, for their exceptional collaboration. The views expressed herein are those of the author(s) and are not to be construed as official or reflecting the views of the Commandant or of the U.S. Coast Guard. This research is supported by US Coast Guard grant HSHQDC-10-D-00019 and MURI grant W911NF-11-1-0332. We also thank the anonymous reviewers for valuable suggestions.

A preliminary version of this work appears as the conference paper (Fang, Jiang, & Tambe, 2013). There are several major advances in this article: (i) Whereas the earlier work confined targets to move in 1-D space, we provide a significant extension of our algorithms (DASS and CASS) in this article to enable the targets and the patrollers to move in 2-D space; we also provide detailed experimental results on this 2-D extension. (ii) We provide additional novel equilibrium refinement approaches and experimentally compare their performance with the equilibrium refinement approach offered in our earlier work; this allows us to offer an improved understanding of the equilibrium refinement space. (iii) We discuss several sampling methods in detail to sample actual patrol routes from the mixed strategies we generate – a discussion that was missing in our earlier work. (iv) We provide detailed proofs that were omitted in the previous version of the work.

Appendix A. Notation Table

Notation	Meaning
MRMT	The problem of multiple Mobile Resources protecting Moving Targets
MRMT_{sg}	Game model with a continuous set of strategies for the attacker for MRMT.
L	Number of ferries.
F_q	Ferry with index q.
A, B	Terminal points.
T	Continuous time interval or a finite set of time points.
D	Continuous space of possible locations or a set of distance points.
$S_q(t)$	Ferry schedule. Position of the target F_q at a specified time t.
W	Number of patrollers.
P_u	Patroller with index u.
v_m	Speed limit of patroller.
r_e	Protection radius of patroller.
C_G	Probability that the attacker can be stopped with G patrollers.
$U_q(t)$	Positive reward of a successful attack on target F_q at time t for the attacker.
M	Number of discretized time points.
N	Number of discretized distance points.
t_k	Discretized time point.
d_i	Discretized distance point.
δt	Distance between two adjacent time points.
R_u	Patrol route for patroller P_u. Under discretization of the defender's strategy space, R_u can be described as a vector.
$r_u(k)$	The patroller is located at $d_{r_u(k)}$ at time t_k.
$f(i,j,k)$	Flow distribution variable. Probability that the patroller moves from d_i to d_j during time $[t_k, t_{k+1}]$.
$p(i,k)$	Marginal distribution variable. Probability that the patroller is located at d_i t_k.
$E_{i,j,k}$	The directed edge linking nodes (t_k, d_i) and (t_{k+1}, d_j).
$p(R_u)$	Probability of taking route R_u.
$\text{AttEU}(F_q, t)$	Attacker expected utility of attacking target F_q at time t.
$\beta_q(t)$	Protection range of target F_q at time t
$\omega(F_q, t)$	Probability that the patroller is protecting target F_q at time t.
$I(i,q,k)$	Whether a patroller located at d_i at time t_k is protecting target F_q.
L_q^1,L_q^2	Lines of $S_q(t) \pm r_e$.
θ_{qk}^r	The rth intersection point in $[t_k, t_{k+1}]$ with respect to target F_q.
$\text{AttEU}(F_q, \theta_{qk}^{r\pm})$	Left/right-side limit of $\text{AttEU}(F_q, t)$ at θ_{qk}^r.
M_{qk}	Number of intersection points in $[t_k, t_{k+1}]$ with respect to target F_q.
$A_{qk}^r(i,j)$	C_1 if patroller taking edge $E_{i,j,k}$ can protect target F_q in $[\theta_{qk}^r, \theta_{qk}^{r+1}]$; 0 otherwise.
$E(u,k)$	Short for $E_{r_u(k), r_u(k+1), k}$.

Table 8: Summary of notations involved in the paper.

Appendix B. Calculation of Intersection Points in CASS for 2-D Settings

We calculate the time where the patroller's route intersects with the protection range for a target when the patroller is moving from $V_1 = (x_1, y_1)$ to $V_2 = (x_2, y_2)$ and the target is moving from $S_q(t_k) = (\hat{x_1}, \hat{y_1})$ to $S_q(t_{k+1}) = (\hat{x_2}, \hat{y_2})$ during $[t_k, t_{k+1}]$. The patroller's position at a given time $t \in [t_k, t_{k+1}]$ is denoted as (x, y) and the target's position is denoted as $(\hat{x}, \hat{y})$. Then we have

$$x = \frac{t - t_k}{t_{k+1} - t_k}(x_2 - x_1) + x_1, \qquad y = \frac{t - t_k}{t_{k+1} - t_k}(y_2 - y_1) + y_1 \tag{40}$$

$$\hat{x} = \frac{t - t_k}{t_{k+1} - t_k}(\hat{x_2} - \hat{x_1}) + x_1, \qquad \hat{y} = \frac{t - t_k}{t_{k+1} - t_k}(\hat{y_2} - \hat{y_1}) + \hat{y_1} \tag{41}$$

At an intersection point, the distance from the patroller's position to the target's position equals to the protection radius r_e, so we are looking for a time t such that

$$(x - \hat{x})^2 + (y - \hat{y})^2 = r_e^2 \tag{42}$$

By substituting the variables in Equation 42 with Equations 40–41, and denoting

$$A_1 = \frac{(x_2 - x_1) - (\hat{x_2} - \hat{x_1})}{t_{k+1} - t_k}, \qquad B_1 = x_1 - \hat{x_1},$$

$$A_2 = \frac{(y_2 - y_1) - (\hat{y_2} - \hat{y_1})}{t_{k+1} - t_k}, \qquad B_2 = y_1 - \hat{y_1},$$

Equation 42 can be simplified to

$$(A_1 t - A_1 t_k + B_1)^2 + (A_2 t - A_2 t_k + B_2)^2 = r_e^2. \tag{43}$$

Denote $C_1 = B_1 - A_1 t_k$ and $C_2 = B_2 - A_2 t_k$, and we can easily get the two roots of this quadratic equation, which are

$$t_{a,b} = \frac{-2(A_1C_1 + A_2C_2) \pm 2\sqrt{(A_1C_1 + A_2C_2)^2 - (A_1^2 + A_2^2)(C_1^2 + C_2^2 - r_e^2)}}{2(A_1^2 + A_2^2)}. \tag{44}$$

t_a or t_b is the time of a valid intersection point if and only if it is within the time interval under consideration ($[t_k, t_{k+1}]$).

References

Agmon, N., Kraus, S., & Kaminka, G. A. (2008). Multi-robot perimeter patrol in adversarial settings. In *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2339–2345.

Alpern, S. (1992). Infiltration Games on Arbitrary Graphs. *Journal of Mathematical Analysis and Applications*, *163*, 286–288.

An, B., Kempe, D., Kiekintveld, C., Shieh, E., Singh, S. P., Tambe, M., & Vorobeychik, Y. (2012). Security games with limited surveillance. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence*, pp. 1241–1248.

An, B., Tambe, M., Ordóñez, F., Shieh, E., & Kiekintveld, C. (2011). Refinement of strong stackelberg equilibria in security games. In *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence (AAAI)*, pp. 587–593.

Basilico, N., Gatti, N., & Amigoni, F. (2009). Leader-follower strategies for robotic patrolling in environments with arbitrary topologies. In *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems (AAMAS) - Volume 1*, pp. 57–64.

Bošanský, B., Lisý, V., Jakob, M., & Pěchouček, M. (2011). Computing time-dependent policies for patrolling games with mobile targets. In *The 10th International Conference on Autonomous Agents and Multiagent Systems (AAMAS) - Volume 3*, pp. 989–996.

Conitzer, V., & Sandholm, T. (2006). Computing the optimal strategy to commit to. In *Proceedings of the 7th ACM Conference on Electronic Commerce*, EC '06, pp. 82–90.

Daskalakis, C., & Weinberg, S. M. (2012). Symmetries and optimal multi-dimensional mechanism design. In *Proceedings of the 13th ACM Conference on Electronic Commerce*, EC '12, pp. 370–387.

Fang, F., Jiang, A. X., & Tambe, M. (2013). Optimal patrol strategy for protecting moving targets with multiple mobile resources. In *Proceedings of the 2013 International Conference on Autonomous Agents and Multi-agent Systems*, AAMAS '13, pp. 957–964.

Fudenberg, D., & Tirole, J. (1991). *Game Theory*. MIT Press.

Gal, S. (1980). *Search Games*. Academic Press, New York.

Gatti, N. (2008). Game theoretical insights in strategic patrolling: Model and algorithm in normal-form. In *Proceedings of the 18th European Conference on Artificial Intelligence (ECAI)*, pp. 403–407.

Greenberg, M., Chalk, P., & Willis, H. (2006). *Maritime terrorism: risk and liability*. Rand Corporation monograph series. RAND Center for Terrorism Risk Management Policy.

Halvorson, E., Conitzer, V., & Parr, R. (2009). Multi-step Multi-sensor Hider-Seeker Games. In *IJCAI*.

Jakob, M., Vaněk, O., & Pěchouček, M. (2011). Using agents to improve international maritime transport security. *Intelligent Systems, IEEE*, *26*(1), 90–96.

Jiang, A. X., Yin, Z., Zhang, C., Tambe, M., & Kraus, S. (2013). Game-theoretic randomization for security patrolling with dynamic execution uncertainty. In *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*, AAMAS '13, pp. 207–214.

Johnson, M. P., Fang, F., & Tambe, M. (2012). Patrol strategies to maximize pristine forest area. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence (AAAI)*, pp. 295–301.

Kiekintveld, C., Islam, T., & Kreinovich, V. (2013). Security games with interval uncertainty. In *Proceedings of the 2013 International Conference on Autonomous Agents and Multi-agent Systems*, AAMAS '13, pp. 231–238.

Kiekintveld, C., Jain, M., Tsai, J., Pita, J., Ordóñez, F., & Tambe, M. (2009). Computing optimal randomized resource allocations for massive security games. In *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems - Volume 1*, AAMAS '09, pp. 689–696.

Korzhyk, D., Conitzer, V., & Parr, R. (2010). Complexity of computing optimal Stackelberg strategies in security resource allocation games. In *Proceedings of the 24th National Conference on Artificial Intelligence (AAAI)*, pp. 805–810.

Korzhyk, D., Conitzer, V., & Parr, R. (2011). Security games with multiple attacker resources. In *Proceedings of the Twenty-Second international joint conference on Artificial Intelligence - Volume Volume One*, IJCAI'11, pp. 273–279. AAAI Press.

Krause, A., Roper, A., & Golovin, D. (2011). Randomized sensing in adversarial environments. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 2133–2139.

Krishna, V. (2009). *Auction theory*. Academic press.

Kumar, A., & Zilberstein, S. (2010). Anytime planning for decentralized POMDPs using expectation maximization. In *Proceedings of the Twenty-Sixth Conference on Uncertainty in Artificial Intelligence*, pp. 294–301.

Letchford, J. (2013). *Computational Aspects of Stackelberg Games*. Ph.D. thesis, Duke University.

Letchford, J., & Conitzer, V. (2013). Solving security games on graphs via marginal probabilities. In *Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence (AAAI)*, pp. 591–597.

Letchford, J., & Vorobeychik, Y. (2012). Computing optimal security strategies for interdependent assets. In *The Conference on Uncertainty in Artificial Intelligence (UAI)*, pp. 459–468.

Luber, S., Yin, Z., Fave, F. D., Jiang, A. X., Tambe, M., & Sullivan, J. P. (2013). Game-theoretic patrol strategies for transit systems: the trusts system and its mobile app (demonstration). In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)[Demonstrations Track]*, pp. 1377–1378.

Marecki, J., Tesauro, G., & Segal, R. (2012). Playing repeated stackelberg games with unknown opponents. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems*, AAMAS '12, pp. 821–828.

Miltersen, P. B., & Sørensen, T. B. (2007). Computing proper equilibria of zero-sum games. In *Proceedings of the 5th International Conference on Computers and Games*, CG'06, pp. 200–211.

Owen, G. (1995). *Game Theory (3rd ed.)*. Academic Press.

Paruchuri, P., Tambe, M., Ordóñez, F., & Kraus, S. (2006). Security in multiagent systems by policy randomization. In *Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, AAMAS '06, pp. 273–280.

Pita, J., Jain, M., Marecki, J., Ordóñez, F., Portway, C., Tambe, M., Western, C., Paruchuri, P., & Kraus, S. (2008). Deployed ARMOR protection: the application of a game theoretic model for security at the Los Angeles International Airport. In *Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems: Industrial Track*, AAMAS '08, pp. 125–132.

Pita, J., Jain, M., Ordonez, F., Portway, C., Tambe, M., Western, C., Paruchuri, P., & Kraus, S. (2009). Using game theory for los angeles airport security.. *AI Magazine*, *30*, 43–57.

Shieh, E., An, B., Yang, R., Tambe, M., Baldwin, C., DiRenzo, J., Maule, B., & Meyer, G. (2012). 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 - Volume 1*, AAMAS '12, pp. 13–20.

Simon, L. K., & Stinchcombe, M. B. (1995). Equilibrium refinement for infinite normal-form games. *Econometrica*, *63*(6), 1421–1443.

Stein, N. D., Ozdaglar, A., & Parrilo, P. A. (2008). Separable and low-rank continuous games. *International Journal of Game Theory*, *37*(4), 475–504.

Stone, P., Kaminka, G. A., Kraus, S., & Rosenschein, J. S. (2010). Ad hoc autonomous agent teams: Collaboration without pre-coordination. In *Proceedings of the 24th AAAI Conference on Artificial Intelligence*, pp. 1504–1509.

Tambe, M. (1997). Towards flexible teamwork. *JOURNAL OF ARTIFICIAL INTELLIGENCE RESEARCH*, *7*, 83–124.

Tambe, M. (2011). *Security and Game Theory: Algorithms, Deployed Systems, Lessons Learned*. Cambridge University Press.

Thompson, D. R. M., & Leyton-Brown, K. (2009). Computational analysis of perfect-information position auctions. In *Proceedings of the 10th ACM conference on Electronic commerce*, EC '09, pp. 51–60.

Tsai, J., Rathi, S., Kiekintveld, C., Ordonez, F., & Tambe, M. (2009). IRIS - a tool for strategic security allocation in transportation networks. In *The Eighth International Conference on Autonomous Agents and Multiagent Systems - Industry Track*, AAMAS '09, pp. 37–44.

van Damme, E. (1987). *Stability and Perfection of Nash equilibria*. Springer-Verlag.

Vaněk, O., Jakob, M., Hrstka, O., & Pěchouček, M. (2011). Using multi-agent simulation to improve the security of maritime transit. In *Proceedings of 12th International Workshop on Multi-Agent-Based Simulation (MABS)*, pp. 1–16.

Vorobeychik, Y., & Singh, S. (2012). Computing stackelberg equilibria in discounted stochastic games. In *Proceedings of the Twenty-Sixth Conference on Artificial Intelligence (AAAI)*, pp. 1478–1484.

Yin, Z., Jiang, A. X., Johnson, M. P., Kiekintveld, C., Leyton-Brown, K., Sandholm, T., Tambe, M., & Sullivan, J. P. (2012). TRUSTS: Scheduling randomized patrols for fare inspection in transit systems. In *Proceedings of the Twenty-Fourth Conference on Innovative Applications of Artificial Intelligence (IAAI)*, pp. 2348–2355.

Yin, Z., & Tambe, M. (2011). Continuous time planning for multiagent teams with temporal constraints. In *Proceedings of the Twenty-Second international joint conference on Artificial Intelligence - Volume Volume One*, IJCAI'11, pp. 465–471. AAAI Press.