Optimal Patrol Strategy for Protecting Moving Targets with Multiple Mobile Resources

Fei Fang, Albert Xin Jiang, Milind Tambe
University of Southern California
Los Angeles, CA, 90089
{feifang,jiangx,tambe}@usc.edu

ABSTRACT

Previous work on Stackelberg Security Games for scheduling security resources 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 lead to a continuous set of strategies for the players. The problem is motivated by several real-world domains including protecting ferries with escorts 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-program-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) a heuristic method of equilibrium refinement for improved robustness and (iv) detailed experimental analysis in the ferry protection domain.

Categories and Subject Descriptors

I.2.11 [**ARTIFICIAL INTELLIGENCE**]: Distributed Artificial Intelligence—*Intelligent agents, Multiagent systems*

General Terms

Security, Algorithms

Keywords

Continuous Strategy Set, Equilibrium Refinement, Moving Targets

1. INTRODUCTION

Stackelberg security games have been successfully deployed in a number of infrastructure security applications [12], most recently in the TRUSTS system in use by the Los Angeles Sheriff's department [13]. In these games, the leader (defender) commits to a randomized schedule – a probability distribution over deterministic schedules – and the follower (attacker) then surveils the distribution and then plays a best response.

This paper focuses on modeling patrolling domains with a mobile set of targets. The attacker can attack these targets at any time during their movement, leading to a continuous set of strategies in the resulting game. The defender has a set of mobile patroller(s) to protect these targets. As opposed to previous work [1, 5, 4], our contributions include computing optimal strategies for the defender while reasoning about the attacker's continuous strategy set *without discretization*. This work further models target values that vary depending on location and time and assumes a zero-sum game. The defender's objective is to schedule the mobile patrol resources to minimize attacker's maximum expected utility.

The first contribution of this paper is a novel game model called MRMT_{sg} for this problem of multiple Mobile Resources protecting Moving Targets. MRMT_{sg} is an attacker-defender Stackelberg game model with a continuous set of strategies for the players, particularly for the attacker. More specifically, while the defender's strategy space is also continuous, we discretize it for three reasons. Firstly, 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 is still valid in the original game with continuous strategy space for the defender, so the solution calculated under our formulation gives a guarantee in terms of expected utility for the original continuous game. On the other hand, discretizing the attacker's strategy space can be highly problematic: if we assume the attacker could only attack at certain descretized 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*, exploiting the piecewise linear nature of the attacker's expected utility function. 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 a heuristic equilibrium refinement approach for the game. CASS is currently being considered for deployment by the US Coast Guard.

2. RELATED WORK

As mentioned before, Stackelberg games have been widely applied to security domains [13]. However, most of this work has considered static targets [12]. Even when the players are mobile, e.g. in hider-seeker games [8], infiltration games [2] or search games [7],

Appears in: *Proceedings of the 12th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2013), Ito, Jonker, Gini, and Shehory (eds.), May, 6–10, 2013, Saint Paul, Minnesota, USA.*

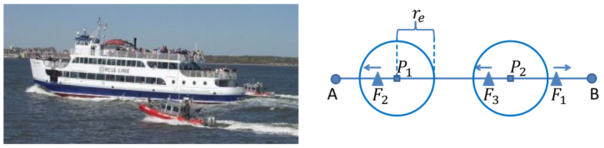
the models have considered static targets if any. Additionally, even when the targets were mobile, e.g., trains [13], the players were restricted to move along the targets to protect or attack them (the targets there are in essence stationary).

With respect to related work computing defender strategies for patrolling domains, Noah et al. [1] compute strategies for setting up a perimeter patrol in adversarial settings with mobile patrollers. Similarly, Basilico et al. [4] compute the leader-follower equilibrium for robotic patrolling in environments with arbitrary topologies. In the same way, Johnson et al. [9] propose a continuous game model for protecting forests from illegal logging. However, in contrast to our problem, the targets are stationary in all this related work.

Bosansky et al. have studied the problem of protecting moving targets [5], similar to our domain. However, they considered a model in which the defender, the attacker and targets have discretized movements on a directed graph. We, in our work, generalize the strategy 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, Bosanky et al. presented a formulation with non-linear constraints that faced scaling problems even with a single defender resource.

3. 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 to attack (e.g., with a small boat packed with explosives that may be only detected once it gets close to the ferry). Small, fast patrol boats can provide protection to such ferries (Figure 1(a)), but there are often limited numbers of patrol boats, i.e., they cannot protect the ferries at all times at all locations. Other examples include protecting refugee aid convoys with overhead UAVs.



(a) (b)

Figure 1: (a) Protecting ferries with patrol boats; (b) Example with three targets (triangles) and two patrollers (squares). Patroller P_1 is protecting F_2 and P_2 is protecting F_3.

Domain description. In this problem, there are L moving targets, $F_1, F_2, ..., F_L$[1]. 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 (an illustrative instance is shown in Figure 1(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. The 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, $T = [0, 1]$ represents the continuous time interval of a typical day (normalized) and $D = [0, 1]$ is the continuous space of possible locations (normalized) with 0 corresponding to terminal A and 1 terminal B. So $S_q(t)$ 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 faster than the targets, they have a maximum speed v_m (range of velocity is $[-v_m, v_m]$). The attacker will choose a certain time and a certain target to attack. 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 in Figure 1(b)). 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 stronger protection.*

As with previous work in security games [12, 13], 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 are currently 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 [12], which we remedy in this paper. (In the rest of the paper, we denote the defender with "she" and the attacker with "he").

Defender strategy. A pure strategy of defender is to designate a moving 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.

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 (F_q, t) where F_q is the target to attack and t is the time to attack.

Utilities. We assume the game is zero-sum. If the attacker performed 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 $U_q(t)$ can be represented as a piecewise linear function of t for each target F_q (we will show an example in Section 6).

4. 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 DASS approach to compute a minimax solution for discretized attacker strategy space (Section 4.2), followed by CASS for the attacker's continuous strategy space (Section 4.3). We first assume a single patroller and then generalize to multiple patrollers in Section 4.4. Since our game is zero-sum, we use minimax (minimizing

[1] A table of all notations can be found in the online appendix (*http://mrmt.webs.com/*).

the maximum attacker utility) which returns the same solution as Strong Stackelberg Equilibrium [6, 10] for MRMT$_{\text{sg}}$.

4.1 Representing Defender's Strategies

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 require δt to be small enough such that for each target F_q, the utility function $U_q(t)$ and the moving schedule $S_q(t)$ are linear within each interval $[t_k, t_{k+1}]$ for $k = 1, \ldots, M-1$, i.e., the target is moving with uniform speed and linearly changing utility during each of these intervals.

In addition to discretization in time, we also discretize the line segment AB[2] that the targets move along 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. 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}. 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 (R_u is, in essence, a mapping of $T \rightarrow D$) can be represented as a vector $R_u = (d_{R_u(1)}, d_{R_u(2)}, ..., d_{R_u(M)})$. $R_u(k)$ is the index of the discretized distance point where the patroller is located at time t_k.

For a single defender resource in the *full representation*, the defender's mixed strategy assigns a probability to each of the patrol routes that can be executed. If v_m is large enough, there are in total N^M patrol routes, which makes the full representation intractable. Therefore, we use a *compact representation* of the defender's mixed strategy where we represent the defender's strategy with *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 $O(MN^2)$, much more efficient compared to the full representation. Figure 2 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 in Yin et al. [13], we use it in a continuous setting.

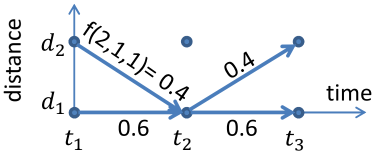


Figure 2: Compact representation: x-axis shows time intervals; y-axis the discretized distance-points in the one-dimensional movement space.

Any strategy in full representation can be mapped into a compact representation. If there are H possible patrol routes $R_1, R_2, ..., R_H$, a strategy in full representation can be denoted 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 $R_u(k+1) = j$. Formally,

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

Different mixed strategies in full representation can be mapped to the same compact representation. Table 1 shows a simple example. Rows 1 and 2 show full representation for two mixed strategies. 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 (shown in Figure 2).

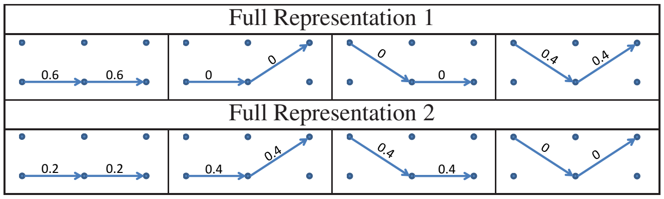


Table 1: Two full representations that can be mapped into the same compact representation shown in Figure 2.

This compact representation does not lead to any loss in solution quality. Recall our goal is to find an optimal defender strategy that minimizes maximum attacker utility. The attacker expected utility of attacking target F_q at time t given defender strategy f can be expressed as

$$\text{AttEU}_f(F_q, t) = (1 - C_1\omega(F_q, t))U_q(t), \tag{2}$$

where $U_q(t)$ is the reward for a successful attack, $\omega(F_q, t)$ 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 (F_q, t), AttEU(F_q, t) is purely decided by $\omega(F_q, t)$. As we will show in the next subsection, $\omega(F_q, t)$ 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 AttEU for any attacker's pure strategy (F_q, t). 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 as* $p(i,k)$, *then* $p(i,k) = \sum_{j=1}^{N} f(j,i,k-1) = \sum_{j=1}^{N} f(i,j,k)$.

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

4.2 DASS: Discretized Attacker Strategies

DASS (Solver for Discretized Attacker Strategies) efficiently finds minimax solutions for MRMT-based games while constraining the attacker to attack at discretized time points t_k. That is, we need to minimize v where v is the maximum of attacker's expected utility. Here, v is the maximum of AttEU(F_q, t) for any target F_q at any discretized time point t_k.

[2] It is possible to have additional points where targets may stop (e.g., to load and unload passengers).

From Equation (2), we know that 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 $\beta(F_q, t) = [\max\{S_q(t) - r_e, d_1\}, \min\{S_q(t) + r_e, d_N\}]$. If the patroller is located within the range $\beta(F_q, t)$, 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 $\beta(F_q, t)$ 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 2. *$I(i, q, k)$ is a function of two values. $I(i, q, k) = 1$ if $d_i \in \beta(F_q, t_k)$, 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 is protecting target F_q. The probability that the patroller is at d_i at time t_k is $p(i, k)$. So we have

$$\omega(F_q, t_k) = \sum\nolimits_{i:I(i,q,k)=1} p(i, k), \tag{3}$$

$$\text{AttEU}(F_q, t_k) = \left(1 - C_1 \sum\nolimits_{i:I(i,q,k)=1} p(i, k)\right) U_q(t). \tag{4}$$

Equation (4) follows from Equations (2) and (3).Finally, we must address speed restrictions on the patroller. We can set all flows that are not achievable to zero, 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:

$$\min_{f(i,j,k),p(i,k)} v \tag{5}$$

$$f(i, j, k) \in [0, 1], \forall i, j, k \tag{6}$$

$$f(i, j, k) = 0, \forall i, j, k \text{ such that } |d_j - d_i| > v_m \delta t \tag{7}$$

$$p(i, k) = \sum_{j=1}^{N} f(j, i, k - 1), \forall i, \forall k > 1 \tag{8}$$

$$p(i, k) = \sum_{j=1}^{N} f(i, j, k), \forall i, \forall k < M \tag{9}$$

$$\sum_{i=1}^{N} p(i, k) = 1, \forall k \tag{10}$$

$$v \geq \text{AttEU}(F_q, t_k), \forall q, \forall k \tag{11}$$

Constraint 7 describes the speed limit. Constraint 6 describes the probability range. Constraints 8–9 describes Property 2. Constraint 10 is exactly Property 3. Property 1 can be derived from the 2 and 3, so it is not listed as a constraint. Equation (11) 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 (4).

4.3 CASS: Continuous Attacker Strategies

Unfortunately, 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: Figure 3 shows the defender's compact strategy between t_k and t_{k+1}. Here the defender's marginal strategy has only three non-zero 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 $E^+ = \{E_{3,4,k}, E_{3,1,k}, E_{1,3,k}\}$. There is only one target, which 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_1. The dark lines L_1^1 and L_1^2 are parallel to S_1 with distance r_e. The area between them indicates the protection range $\beta(F_q, t)$ for any time $t \in (t_k, t_{k+1})$. Consider the time points at which an edge from E^+ intersects one of L_1^1, L_1^2 (labeled as $\theta_k^r, r = 1 \ldots 4$ in Figure 3). 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_k^0 and t_{k+1} as θ_k^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_k^0 to θ_k^1 because $E_{3,4,k}$ is between L_1^1 and L_1^2 in $[\theta_k^0, \theta_k^1]$, during which the distance to the target is less or equal than protection radius r_e. Consider the sub-intervals between each θ_k^j and θ_k^{j+1}, for $j = 0 \ldots 4$. 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 4(a).

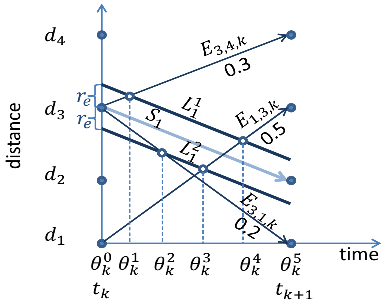


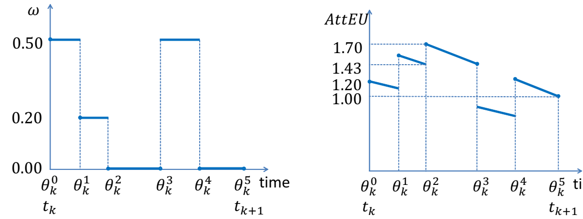
Figure 3: Changes of AttEU in (t_k, t_{k+1}).

Suppose $U(F_q, t)$ 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 AttEU(F_q, t) for (t_k, t_{k+1}), as plotted in Figure 4(b). AttEU(F_q, t) is linear in each sub-interval but the function is discontinuous at the intersection points $\theta_k^1, \ldots, \theta_k^4$ because of the discontinuity of $\omega(F_q, t)$. We denote:

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

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

An attacker can choose to attack at a time immediately after θ_k^2, getting an expected utility that is arbitrarily close to 1.70. According to Equation (4), we can get AttEU$(F_q, t_k) = 1.20$ and AttEU$(F_q, t_{k+1}) = 1.00$, both lower than AttEU(F_q, θ_k^{2+}).



(a) Probability that the target is protected in (t_k, t_{k+1}).

(b) The attacker's expected utility in (t_k, t_{k+1}).

Figure 4: Sub-interval analysis

Thus, the attacker can get a higher expected reward by attacking between t_k and t_{k+1}. 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 AttEU(F_q, t). Supremum is defined to be the smallest real number that is greater than or equal to any AttEU(F_q, t), i.e., it is the least upper bound. In the above example, the supremum of attacker's expected utility in (t_k, t_{k+1}) is AttEU$(F_q, \theta_k^{1+}) = 1.70$. Formally, a defender strategy f is minimax if $f \in \arg\min_{f'} \sup \text{AttEU}_{f'}(F_q, t)$.

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 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 intersection points of edges $E_{i,j,k}$ and L_q^1, L_q^2. We sort the intersection points in increasing order, denoted as $\theta_k^r, r = 1 \ldots M_{qk}$, where M_{qk} is the total number of intersection points. Set $\theta_k^0 = t^k$ and $\theta_k^{M_{qk}+1} = t^{k+1}$. Thus (t_k, t_{k+1}) is divided into sub-intervals $(\theta_k^r, \theta_k^{r+1}), r = 0, ..., M_{qk}$.

LEMMA 1. *AttEU(F_q, t) is piecewise linear in t with the same set of potential discontinuous points independent of the defender strategy.*

The proof is in the online appendix(*http://mrmt.webs.com/*) and follows from the fact that in each sub-interval, $E_{i,j,k}$ is either totally above or below L_q^1, and similarly for L_q^2. Define coefficient $A_{qk}^r(i,j)$ to be C_1 if edge $E_{i,j,k}$ is between L_q^1 and L_q^2 in $(\theta_k^r, \theta_k^{r+1})$, and 0 otherwise. According to Equation (2) and the fact that $\omega(F_q, t)$ is the sum of $f(i,j,k)$ whose corresponding coefficient $A_{qk}^r(i,j) = C_1$, we have the following equation for $t \in (\theta_k^r, \theta_k^{r+1})$.

$$\text{AttEU}(F_q, t) = \left(1 - \sum_{i=1}^{N} \sum_{j=1}^{N} A_{qk}^r(i,j) f(i,j,k)\right) \cdot U_q(t) \quad (12)$$

Piecewise linearity of AttEU(F_q, t) means the function is monotonous in each sub-interval and the supremum can be found at the intersection points. Because of linearity, the supremum of AttEU in $(\theta_k^r, \theta_k^{r+1})$ can only be chosen from the one-sided limits of the endpoints, AttEU(F_q, θ_k^{r+}) and AttEU$(F_q, \theta_k^{(r+1)-})$. Furthermore, if $U(F_q, t)$ is decreasing in $[t_k, t_{k+1}]$, the supremum is AttEU(F_q, θ_k^{r+}) and otherwise it is AttEU$(F_q, \theta_k^{(r+1)-})$. In other words, all other attacker's strategies in $(\theta_k^r, \theta_k^{r+1})$ are dominated by attacking at time close to θ_k^r or θ_k^{r+1}. Thus, CASS adds new constraints to Constraints 6–11 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:

$$\min_{f(i,j,k), p(i,k)} v \quad (13)$$

$$\text{subject to constraints}(6..11)$$

$$v \geq \max\{\text{AttEU}(F_q, \theta_k^{r+}), \text{AttEU}(F_q, \theta_k^{(r+1)-})\} \quad (14)$$
$$\forall k = 1..M, q = 1..L, r = 0..M_{qk}$$

This linear program stands at the core of CASSÀll the linear constraints included by Constraint 14 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 CalInt(L_q^1, L_q^2, v_m) in Line 5 returns the list of all intersection time points between all possible edges $E_{i,j,k}$ and the parallel lines L_q^1, L_q^2, with additional points t_k as θ_k^0 and t_{k+1} as $\theta_k^{M_{qk}+1}$. Function CalCoef$(L_q^1, L_q^2, v_m, \theta_k^r, \theta_k^{r+1})$ in Line 7 returns the coefficient matrix A_{qk}^r. Lines 8–11 add a constraint with respect to the larger value of AttEU(F_q, θ_k^{r+}) and AttEU$(F_q, \theta_k^{(r+1)-})$ to CASS for this sub-interval $(\theta_k^r, \theta_k^{r+1})$.

THEOREM 1. *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, AttEU(F_q, t) is piecewise linear and discontinuity can only occur at the intersection points θ_k^r. These intersection points divide the time space into sub-intervals. Because of piecewise linearity, the supremum of 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, v is no less than any of the limit values at the intersection points according to Constraint 14, and thus v can be any upper bound of AttEU(F_q, t) for f. As v is minimized in the objective function, v is no greater than the supremum of AttEU(F_q, t) given any defender strategy f, and further v will be the infimum of the set of supremum corresponding to all defender strategies. Thus we get the optimal defender strategy.

The total number of variables in the linear program is $O(MN^2)$ (including $f(i,j,k)$ and $p(i,k)$). The number of constraints represented in Algorithm 1 is $O(MN^2)$ as the number of intersection points is at most $2(M-1)N^2$. The number of constraints represented in Constraints 6–11 is also $O(MN^2)$. 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.*

Algorithm 1: Add constraints described in Constraint 14

```
1  Input: $S_q$, $r_e$, $v_m$, $\{t_k\}$, $\{d_i\}$;
2  for $k \leftarrow 1, \ldots, M-1$ do
3      for $q \leftarrow 1, \ldots, L$ do
4          $L_q^1 \leftarrow S_q + r_e$, $L_q^2 \leftarrow S_q - r_e$;
5          $\theta_k^0, \ldots, \theta_k^{M_{qk}+1} \leftarrow$ CalInt$(L_q^1, L_q^2, v_m)$;
6          for $r \leftarrow 0, \ldots, M_{qk}$ do
7              $A_{qk}^r \leftarrow$ CalCoef$(L_q^1, L_q^2, v_m, \theta_k^r, \theta_k^{r+1})$;
8              if $U(F_q, t)$ is decreasing in $[t_k, t_{k+1}]$ then
9                  add constraint $v \geq$ AttEU$(F_q, \theta_k^{r+})$;
10             else
11                 add constraint $v \geq$ AttEU$(F_q, \theta_k^{(r+1)-})$
```

4.4 Generalized Model

To illustrate generalization to the multiple defender resources case, we take two patrollers as an example. If there are two patrollers, the patrol strategy can be represented as $\{f(i_1, j_1, i_2, j_2, k)\}$. $f(i_1, j_1, i_2, j_2, k)$ shows the 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 2 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 similar to Algorithm 1, the main difference is to 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.

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

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 and get the following equations.

$$\text{AttEU}(F_q, t) = \left(1 - \sum_{Q=1}^{W} C_Q \cdot \omega_Q(F_q, t)\right) \cdot U_q(t)$$

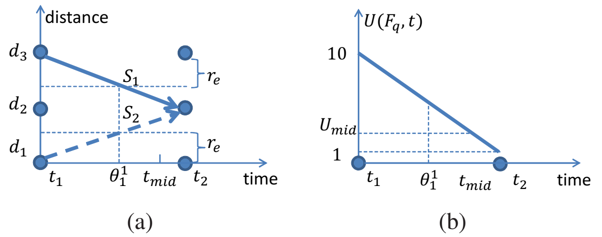
$$\omega_Q(F_q, t_k) = \sum\nolimits_{i_1,\ldots,i_W : \sum\limits_{u=1}^{W} I(i_u,q,k)=Q} p(i_1, \ldots, i_W, k)$$

Q is the number of patrollers protecting the target, and is the probability of protection for the discretized time points t_k. We can modify Algorithm 1 to apply for 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.

The linear program for multiple patrollers can be found in online appendix(*http://mrmt.webs.com/*). The number of variables in the linear program and the number of constraints are both $O(MN^{2W})$. While the expression grows exponentially in the number of resources, in real-world domains such as ferry protection, the number of defender resources are limited. That is the main reason that optimization using security games becomes critical. As a result, the above generalization of CASS is adequate. Indeed, we can easily run CASS with 4 defender resources within 3 minutes for complex ferry domains. Further scale-up is an issue for future work.

5. 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 is moving from d_3 to d_2 and the other is moving from d_1 to d_2 (See Figure 5(a)). $d_3 - d_2 = d_2 - d_1 = d$ and $5d/9 < r_e < d$. 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 5(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_{mid}, he will choose to attack at t_{mid}, and his expected utility is $U(F_q, t_{mid})/2$ for the first equilibrium and $U(F_q, t_{mid})$ for the second. That is, the defender strategy in the first equilibrium is better than the one in the second.



(a) (b)

Figure 5: An example to show different equilibria for one game

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 [6, 11]. For finite security games An *et al* [3] proposed techniques that provide refinement over Stackelberg equilibrium. However there has been little prior research on the computation of equilibrium refinements for continuous games.

We introduce "route-adjust", a heuristic method for refining the equilibrium found by CASS. For expository simplicity, we first consider the single resource case. We begin with defining dominance of defender strategies for MRMT$_{\text{sg}}$.

DEFINITION 3. *Defender strategy f* ***dominates*** *f' if $DefEU_f(F_q, t) \geq DefEU_{f'}(F_q, t)$, $\forall q, t$, or equivalently in this zero-sum game, $AttEU_f(F_q, t) \leq AttEU_{f'}(F_q, t)$.*

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

Corollary 2 follows from Equation (2). Starting with a defender strategy f^0 calculated by CASS, 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. 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 6, 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 we subtract the route from the original flow distribution to get a residual graph. We continue to extract routes from the residual graph until there is no route left. Assume in the flow distribution graph, the number of non-zero terms is Z, 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.

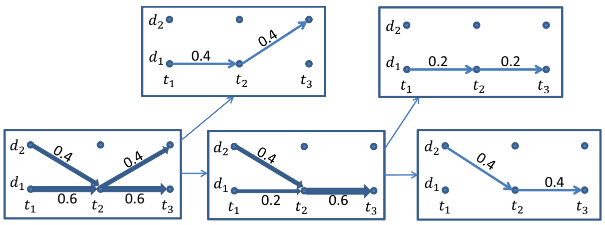


Figure 6: An example of decomposition.

For step (ii), we adjust each of the routes greedily. To that end, we first introduce the (weak) dominance relation of edges and routes, using the intersection points θ_k^r and the coefficient matrix $A_{qk}^r(i,j)$ defined in Section 4.3.

DEFINITION 4. *Edge $E_{i,j,k}$ **dominates** edge $E_{i',j',k}$ in $[t_k, t_{k+1}]$ if $A_{qk}^r(i,j) \geq A_{qk}^r(i',j')$, $\forall q = 1..L$, $\forall r = 1..M_{qk}$, i.e., edge $E_{i,j,k}$ protects target F_q in $[\theta_k^r, \theta_k^{r+1}]$ if edge $E_{i',j',k}$ protects it.*

DEFINITION 5. *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 $E_{R_u(k),R_u(k+1),k}$ dominates edge $E_{R_{u'}(k),R_{u'}(k+1),k}$, $\forall k = 1 \ldots M-1$, i.e., route R_u dominates $R_{u'}$ if each edge of R_u dominates the corresponding edge in $R_{u'}$.*

Denote the original route to be adjusted as R_u and the new route as R_{u_1}. A greedy way to improve the route is to replace only one node in the route, i.e., $R_{u_1}(k) = R_u(k)$, $\forall k \neq k^*$ and $d_{R_u(k^*)}$ in the original route is replaced with $d_{R_{u_1}(k^*)}$. So the patroller's route changes only in $[t_{k^*-1}, t_{k^*+1}]$. To simplify the notation, we denote the edge $E_{R_u(k^*),R_u(k^*+1),k^*}$ as $E(u,k^*)$. Thus, only edges $E(u,k^*-1)$ and $E(u,k^*)$ in the original route are replaced by $E(u_1,k^*-1)$ and $E(u_1,k^*)$ in the new route. We want R_{u_1} to provide more protection to the targets, so the new route should dominate the original one. So for a specified k^*, we are looking for a position $d_{R_{u_1}(k^*)}$ such that: 1) $E(u_1,k^*-1)$ and $E(u_1,k^*)$ meet the speed constraint; 2) $E(u_1,k^*-1)$ and $E(u_1,k^*)$ dominates $E(u,k^*-1)$ and $E(u,k^*)$ respectively; 3) edge $E(u_1,k^*-1)$ and $E(u_1,k^*)$ are not dominated by the corresponding edges for any other choice of $d_{R_{u_1}(k^*)}$. The second requirement ensures the changed edges in the new route dominates the corresponding edges in the original route. Thus the new route R_{u_1} dominates R_u. The third requirement attains a local maxima. We iterate this process, converging to a final route denoted $R_{u'}$.

For the example above, assume the 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 improve each route 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 $E_{i,j,2}$ may be changed. The adjustment is shown in Table 2.

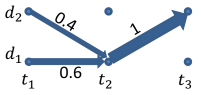
Original	Adjusted	$p(R_u)$		Composed
(d_1, d_1, d_1) (d_1, d_1, d_2) (d_2, d_1, d_1) (d_2, d_1, d_2)	(d_1, d_1, d_2) (d_1, d_1, d_2) (d_2, d_1, d_2) -	0.2 0.4 0.4 0		 d_2 d_1 0.4 1 0.6 t_1 t_2 t_3

Table 2: An example to show how the routes are adjusted

For step (iii), we reconstruct a new compact representation as in Equation 1. For the example above, the result is shown in Table 2.

THEOREM 2. *After steps (i)–(iii), we get a new defender strategy f^1 that dominates the original one f^0.*

While step (iii) allows us to prove Theorem 2, 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, we 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 $A_{qk}^r(i_1, j_1, ..., i_W, j_W)$.

6. EVALUATION

We use an example set in the ferry protection domain and compare the performance in terms of the attacker's expected utility 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.

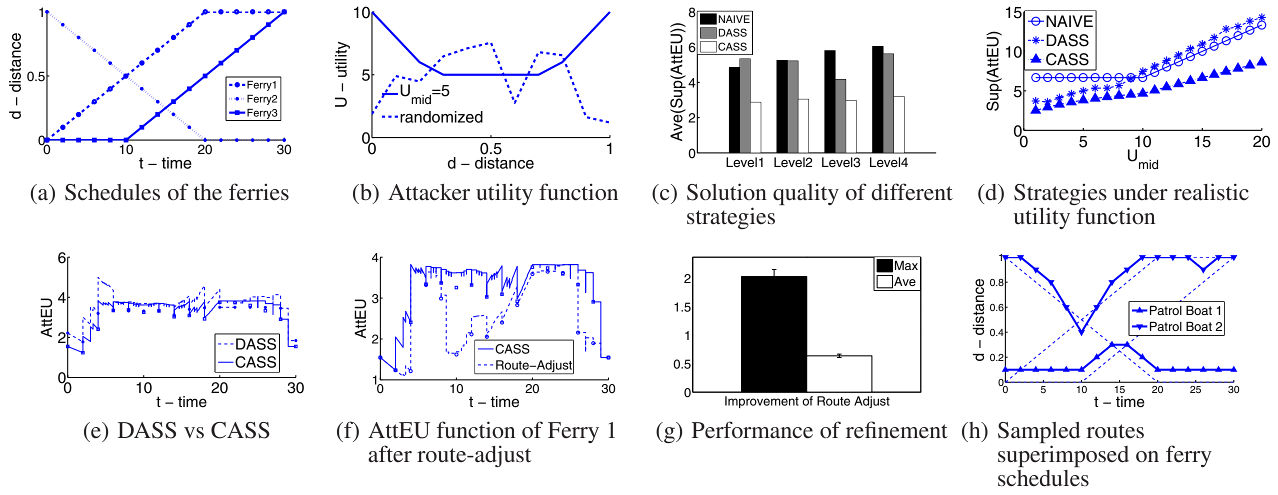
We used the following setting for the experiments, illustrating that this is a complex spatio-temporal game; rather than a discrete security game as in most previous work. There are three ferries moving between terminals A and B and the total distance $AB = 1$. The simulation time is 30 *minutes*. The schedules of the ferries are shown in Figure 7(a), 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. We first show results with 2 patrollers (where $C_1 = 0.8$, and $C_2 = 1.0$), and with more patrollers later.

Performance of CASS. 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 (e.g., at level 1, $M = 4$, $N = 3$, and at level 4, $M = 16$, and $N = 11$) with random utilities, and at each discretization level, we created 20 problem instances. Each instance has utilities uniformly randomly chosen between $[0, 10]$ at discretized points; an example is shown in dashed lines of Figure 7(b). 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 7(c), x-axis plots the four discretization levels. y-axis plots the average attacker utility over the 20 instances for baseline, DASS and CASS. CASS is shown to outperform DASS and baseline (p<0.01).

Next we turn to *more realistic* utility function in this ferry domain, which is of U-shape or inverse U-shape. The solid line in Figure 7(b) shows a sample utility curve where the attacker gains higher utility closer to the shore. In Figure 7(d), we fix the utility at the shore as 10, vary the utility in the middle (the floor of the U-shape or the top of the inverse U-shape), shown on x-axis and compare performance of the strategies in terms of attacker utility 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. Figure 7(e) gives a more detailed analysis for the one instance (shown in Figure 7(b) with solid line). 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(AttEU$(F_1, 20)$), however, the supremum of AttEU(F_1, t), $t \in [0, 30]$ can be as high as 4.99 (AttEU$(F_1, 4^+)$), 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.

Number of Patrollers. Figure 8(a) shows the improvement in performance of CASS with increasing number of patrollers. The x-axis shows the number of patrollers and the y-axis indicates the average of supremum of attacker's expected utility. The results are averaged over the 20 random utility settings for discretization level 1 from Figure 7(c) (only 1 level is shown due to space limit). With less patrollers, the performance of the defender varies a lot depending on the randomized utility function (higher variance). But the variance gets much smaller with more patrollers, which means the defender has sufficient resources for different instances. Figure 8(b) shows the run-time for CASS.

Improvement Using Refinement Methods. In Figure 7(f), we compare the AttEU(F_q, t) function of the defender strategy given by CASS and the defender strategy after equilibrium refinement for Ferry 1 for just one instance described in Figure 7(b)(solid line). In



(a) Schedules of the ferries

(b) Attacker utility function

(c) Solution quality of different strategies

(d) Strategies under realistic utility function

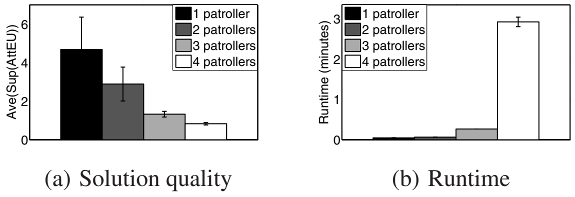
(e) DASS vs CASS

(f) AttEU function of Ferry 1 after route-adjust

(g) Performance of refinement

(h) Sampled routes superimposed on ferry schedules

Figure 7: Experimental settings and results



(a) Solution quality

(b) Runtime

Figure 8: Performance with varying number of patrollers.

route-adjust, we replace every node of the route one-by-one and repeat the process five times. The figure shows for an attack aiming at any target at any time, the defender strategy after refinement is equally good or better than the one in the original equilibrium. Figure 7(g) shows the maximum and the average improvement of refinement over 15 randomized instances for Figure 7(c), which confirms the refinement eliminates the weakly dominated defender strategy calculated by CASS and provides a better defender strategy in terms of possible constrained attackers.

Sampled Routes. We now give a pair of sampled routes for two patrollers for the defender strategy after route-adjust step (ii)(See Figure 7(h)). The x-axis indicates the time and the y-axis is the distance to terminal A. The solid lines show the escorts' patrol routes and the dashed lines show the ferries' schedules.

7. SUMMARY

This paper makes several contributions in computing optimal strategies given moving targets and mobile patrollers: (i) MRMT$_{sg}$, a game model with continuous attacker strategy set; (ii) a fast solution approach, CASS, based on compact representation and subinterval analysis; and (iii) a heuristic method for equilibrium refinement for CASS's solutions; and (iv) detailed experimental analysis in the ferry protection domain. CASS is currently being considered for deployment by the US Coast Guard.

8. ACKNOWLEDGEMENTS

We thank the USCG officers, and particularly Craig Baldwin, Joe Direnzo and Francis Varrichio at sector New York, for their exceptional collaboration. This research is supported by US Coast Guard grant HSHQDC-10-D-00019 and MURI grant W911NF-11-1-0332.

9. REFERENCES

[1] N. Agmon, S. Kraus, and G. A. Kaminka. Multi-robot perimeter patrol in adversarial settings. In *ICRA*, 2008.

[2] S. Alpern. Infiltration Games on Arbitrary Graphs. *Journal of Mathematical Analysis and Applications*, 163:286–288, 1992.

[3] B. An, M. Tambe, F. Ordóñez, E. Shieh, and C. Kiekintveld. Refinement of strong stackelberg equilibria in security games. In *AAAI*, 2011.

[4] N. Basilico, N. Gatti, and F. Amigoni. Leader-follower strategies for robotic patrolling in environments with arbitrary topologies. In *AAMAS*, 2009.

[5] B. Bošanský, V. Lisý, M. Jakob, and M. Pěchouček. Computing time-dependent policies for patrolling games with mobile targets. In *AAMAS*, 2011.

[6] D. Fudenberg and J. Tirole. *Game Theory*. MIT Press, 1991.

[7] S. Gal. *Search Games*. Academic Press, New York, 1980.

[8] E. Halvorson, V. Conitzer, and R. Parr. Multi-step Multi-sensor Hider-Seeker Games. In *IJCAI*, pages 159–166, 2009.

[9] M. P. Johnson, F. Fang, and M. Tambe. Patrol strategies to maximize pristine forest area. In *AAAI*, 2012.

[10] D. Korzhyk, V. Conitzer, and R. Parr. Complexity of computing optimal stackelberg strategies in security resource allocation games. In *AAAI*, pages 805–810, 2010.

[11] P. B. Miltersen and T. B. Sørensen. Computing proper equilibria of zero-sum games. In *Proceedings of the 5th international conference on Computers and games*, CG'06, pages 200–211, Berlin, Heidelberg, 2007. Springer-Verlag.

[12] M. Tambe. *Security and Game Theory: Algorithms, Deployed Systems, Lessons Learned*. Cambridge University Press, 2011.

[13] Z. Yin, A. Jiang, M. Johnson, M. Tambe, C. Kiekintveld, K. Leyton-Brown, T. Sandholm, and J. Sullivan. TRUSTS: Scheduling randomized patrols for fare inspection in transit systems. In *IAAI*, 2012.