Optimal Patrol Planning for Green Security Games with Black-Box Attackers

Haifeng Xu*[1], Benjamin Ford[1], Fei Fang[2], Bistra Dilkina[3], Andrew Plumptre[4], Milind Tambe[1], Margaret Driciru[5], Fred Wanyama[5], Aggrey Rwetsiba[5], Mustapha Nsubaga[6], and Joshua Mabonga[6]

[1] University of Southern California, {haifengx,benjamif,tambe}@usc.edu
[2] Carnegie Mellon University, feifang@cmu.edu
[3] Georgia Institute of Technology, Atlanta, GA, bdilkina@cc.gatech.edu
[4] Wildlife Conservation Society, New York City, NY, aplumptre@wcs.org
[5] Uganda Wildlife Authority, Kampala, Uganda, {margaret.driciru,fred.wanyama,aggrey.rwetsiba}@ugandawildlife.org
[6] Wildlife Conservation Society, Kampala, Uganda, {mnsubuga,jmabonga}@wcs.org

Abstract. Motivated by the problem of protecting endangered animals, there has been a surge of interests in optimizing patrol planning for conservation area protection. Previous efforts in these domains have mostly focused on optimizing patrol routes against a specific boundedly rational poacher behavior model that describes poachers' choices of areas to attack. However, these planning algorithms do not apply to other poaching prediction models, particularly, those complex machine learning models which are recently shown to provide better prediction than traditional bounded-rationality-based models. Moreover, previous patrol planning algorithms do not handle the important concern whereby poachers infer the patrol routes by partially monitoring the rangers' movements. In this paper, we propose OPERA, a general patrol planning framework that: (1) generates optimal implementable patrolling routes against a black-box attacker which can represent a wide range of poaching prediction models; (2) incorporates entropy maximization to ensure that the generated routes are more unpredictable and robust to poachers' partial monitoring. Our experiments on a real-world dataset from Uganda's Queen Elizabeth Protected Area (QEPA) show that OPERA results in better defender utility, more efficient coverage of the area and more unpredictability than benchmark algorithms and the past routes used by rangers at QEPA.

1 Introduction

Worldwide, wildlife conservation agencies have established protected areas to protect threatened species from dire levels of poaching. Unfortunately, even in many protected areas, species' populations are still in decline [3, 1]. These areas are protected by park rangers who conduct patrols to protect wildlife and

* Haifeng Xu and Benjamin Ford are both first authors of this paper.

deter poaching. Given that these areas are vast, however, agencies do not have sufficient resources to ensure rangers can adequately protect the entire park.

At many protected areas, rangers collect observational data while on patrol, and these observations on animals and illegal human activities (e.g., poaching, trespassing) are commonly recorded into a park-wide database (e.g., SMART, Cybertracker). Once enough patrols have been conducted, a patrol manager will analyze the data and generate a new patrolling strategy to execute. However, given the vast area and limited financial budgets of conservation agencies, improving the efficiency of ranger patrols is an important goal in this domain.

Following the success of automated planning tools used in domains such as fare enforcement and seaport protection [20, 15], novel planning tools have also been proposed and applied to ranger patrol planning. Work in [11] developed a new game-theoretic model that optimized against its proposed poacher behavior model to generate randomized patrol strategies. However, they did not account for spatial constraints (i.e., are two areas adjacent?) in their planning and thus cannot guarantee the implementability of their proposed strategies. Moreover, the planning in [11] is specific to one poacher behavior model and cannot be applied to different predictive models. [1] demonstrated the potential for automated planning tools in the real world via a successful field test. However, the planning process in [1] is deterministic and thus is predictable to poachers.

In this paper, we present OPERA (Optimal patrol Planning with Enhanced RAndomness), a general patrol planning framework with the following key features. First, OPERA optimally generates patrols against a black-box poaching prediction model. Unlike other approaches in this domain that can only optimize against their specified prediction model [11, 7], OPERA is capable of optimizing against a wide range of prediction models. Second, OPERA optimizes directly over the space of feasible patrol routes and guarantees implementability of any generated patrol strategy. Lastly, OPERA incorporates entropy maximization in its optimization process to ensure that the generated strategies are sufficiently randomized and robust to partial information leakage – i.e., a frequently observed phenomenon in practice whereby poachers try to infer the patroller's patrolling route by monitoring part of the patroller's movements [13, 10, 18].

We evaluate OPERA on a real-world data set from Uganda's Queen Elizabeth Protected Area (QEPA). Our experiments show that, compared to benchmark heuristic planning algorithms, OPERA results in significantly better defender utility and more efficient coverage of the area. Moreover, the experiments also show that the new entropy maximization procedure results in patrol routes that are much more unpredictable than those routes generated by classical techniques. This effectively mitigates the issue of partial information leakage. Finally, we integrate OPERA with a predictive model of a bagging ensemble of decision trees to generate patrolling routes for QEPA, and compare these routes with the past routes used by rangers at QEPA. The experiments show that OPERA is able to detect all the attacks that are found by past ranger patrolling and also predicted by the predictive model. Moreover, OPERA results in better attack detection and more efficient coverage of the area than the past ranger routes.

2 Related Work

Prior work in planning wildlife patrols has also generated patrol strategies based on a predictive model [11]. In [11], poacher behavior was modeled via a two-layered graphical model and a randomized patrolling strategy was planned in a Stackelberg Security Game (SSG) framework. Similarly, SSGs have been applied to the problem of interdicting rhino poachers [7]. In [7], optimal interdiction strategies were generated for rangers by solving an SSG. However, patrol strategies generated in [11] were not guaranteed to be implementable in the form of patrol routes that satisfy spatial constraints, while [7] optimized over a very small set of patrols specified a priori. In contrast, our scalable approach optimizes over the space of all feasible patrol routes and is guaranteed to generate executable routes. Additionally, both the patrol strategy generation approaches in [11] and [7] were constrained to each of their own adversary behavior models, while our black-box approach can generate a patrol strategy for a wide range of adversary frameworks and corresponding behavior models.

Green Security Games [19, 5] have been introduced to model the interaction in domains such as wildlife protection, fishery protection, and forest protection. In [5], a multi-stage game is used to model the repeated interactions in these domains. In the case where the defender's strategy in one stage can affect the attackers' behavior in future stages, look-ahead planning algorithms were proposed [5] to compute a sequence of defender strategies against attackers that follow a specific behavior model. OPERA can also handle multi-stage planning to generate a sequence of strategies to use, but OPERA additionally introduces a novel and scalable approach to handle black-box attackers.

Other work in this domain has resulted in the successful field testing of planned patrols [1, 4]. [1] generates patrol strategy by reorganizing historical patrol effort values such that areas of highest predicted activity would receive the highest patrol effort. However, such reorganization leads to a deterministic patrol strategy that can be easily exploited by poachers. [4] introduced a patrol planning tool that incorporated spatial constraints to plan detailed patrol routes. However, it relied on a specific type of attacker behavior model [12] while OPERA can optimize against any black-box attacker model that can be approximated by a piece-wise linear function of the patrolling effort.

3 Green Security Games with Black-Box Attackers

In this section, we provide an overview of Green Security Games (GSGs) and how they can work with a black-box attacker model.

3.1 Green Security Games

GSGs are security games that specifically focus on the unique challenges present in conservation domains (e.g., protecting wildlife, fisheries); GSGs focus on protecting threatened natural resources, with limited defender capacity, from repeated outside attacks. Like most of the previous work in GSGs [19, 5], we

consider a discrete setting where the conservation area is divided into N discrete grid cells, each treated as a *target*. Let $[N]$ denote the set of all cells, among which one cell is designated as the *patrol post*. Any patrol route must originate from and return to the patrol post. W.l.o.g., we will treat cell 1 as the patrol post throughout the paper. There is one patroller resource (e.g., ranger team) who patrols $[N]$ cells each day. Due to real-world spatial constraints, from one cell the patroller can only move to neighboring cells. We assume that traversing each cell requires one unit of time, and we let T denote the upper bound of the total units of time that the patroller can patrol each day. As a result, the patroller can traverse at most T cells each day. These spatial constraints can be captured by a time-unrolled graph G (e.g., Figure 1). Any node $v_{t,i}$ in G denotes the cell i at time t. The edges in G, only connecting two consecutive time steps, indicate feasible spatial traversals from one cell to another within a unit of time. Recall that cell 1 is the patrol post, so a feasible patrol route will be a path in G starting from $v_{1,1}$ and ending at $v_{T,1}$ (e.g., the dotted path in Figure 1).

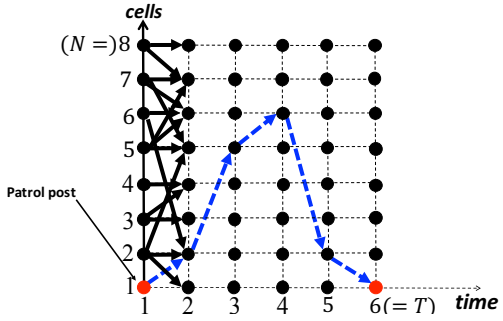

Fig. 1. An Example of the Time-Unrolled Graph

3.2 Black-Box Attackers

Unlike previous security game models which explicitly assume an attacker behavior model (rational or boundedly rational), we assume that for each cell i there is a black-box function g_i, which takes as input certain measure of the defender's patrolling effort l_i^1, l_i^0 at the current and previous period respectively, and outputs a prediction of attacks at cell i. Note that the dependence of the prediction on static features (e.g., animal density, distance & terrain features) is integrated into the function form of g_i and thus will not be an explicit input into g_i. This is motivated by the recent surge of research efforts in using machine learning models to predict attacks in conservation areas [19, 8, 9]. Each of these models can be treated as a black-box function of the attacker's behavior, though we note that the function g_i can also capture perfectly rational or other models of boundedly rational attackers. In this paper, we assume that the function g_i outputs the predicted number of *detected attacks* at cell i (i.e., attacks that happen and are also detected by the patroller) since the real-world data and

corresponding machine learning model we use fit this task. However, we remark that our framework and algorithms are also applicable to other forms of g_i.

We wish to optimize the design of patrol routes against such black-box attackers. Of course, one cannot hope to develop a general, efficient algorithm that works for an arbitrary function g_i. We thus make the following assumption: g_i depends *discretely* on the *patrolling effort* at cell i. More specifically, we assume that there are $m+1$ levels of patrolling effort at any time period, ranging increasingly from 0 to m, and $g_i : \{0, 1, ..., m\}^2 \to \mathbb{R}$ takes level l_i^1, l_i^0 as input. We remark that this discretization is a natural choice since it can be viewed as a *piecewise constant* approximation for a continuous attacker behavior model. It's worth noting that some machine learning models in this domain indeed use discrete patrolling levels as input features [9]. The output of g_i can be any number (e.g., 0–1 for classifiers, a real number for regression).

3.3 Patrolling Effort and its Implementation

Recall that each patrol route is an $s - d$ path in G for $s = v_{1,1}$ and $d = v_{T,1}$. Equivalently, and crucially, a patrol route can also be viewed as a one unit *integer flow* from s to d (e.g., the path in Figure 1 is also a one-unit $s-d$ flow). We allow the defender to randomize her choice of patrol routes, which is called a *defender mixed strategy* and corresponds to a fractional $s - d$ flow. With any period, the patrolling effort x_i at each cell i is the expected total amount of time units spent at cell i during T time steps. For example, using the path in Figure 1, the effort x_2 equals 2 since the path visits cell 2 twice and the efforts $x_5 = 1, x_7 = 0$, etc. When a mixed strategy is used, the effort will be the expected time units. Let $\mathbf{x} = (x_1, ..., x_N)$ denote the patrolling effort vector, or effort vector for short. One important property of the patrolling effort quantity is that it is additive across different time steps. Such additivity allows us to characterize feasible effort vectors using a flow-based formulation. An effort vector is *implementable* if there exists a mixed strategy that results in the effort vector.

Lemma 1. *The effort vector $(x_1, ..., x_N) \in \mathbb{R}_+^N$ within any period is implementable if and only if there exists a flow f in G such that*

$$\begin{array}{ll} x_i = \sum_{t=1}^{T} \left[\sum_{e \in \sigma^+(v_{t,i})} f(e) \right], & \text{for } i = 1, ..., N. \\ f \text{ is a feasible 1-unit } s - d \text{ flow in } G & \end{array} \tag{1}$$

where $\sigma^+(v_{t,i})$ is the set of all edges entering node $v_{t,i}$.

Proof. This simply follows from the observation that any mixed strategy is a 1-unit fractional $s - d$ flow and the definition that x_i is the aggregated effort at cell i from all the T time steps. □

Let $\alpha_0 < \alpha_1 ... < \alpha_m < \alpha_{m+1}$ be $m + 2$ threshold constants which determine the patrol level of any patrolling effort. By default, we always set $\alpha_0 = 0$ and $\alpha_{m+1} = +\infty$. The patrolling effort x_i has level $l \in \{0, 1, ..., m\}$ if $x_i \in [\alpha_l, \alpha_{l+1})$. In most applications, these thresholds are usually given together with the function $g_i(l_i^1, l_i^0)$.

4 Patrolling Route Design to Maximize Attack Detection

Recall that the black-box function g_i in our setting predicts the number of detected attacks at cell i. In this section, we look to design the optimal (possibly randomized) patrol route so that the induced patrol effort maximizes the *total number of detected attacks* $\sum_{i\in[N]} g_i(l_i^1, l_i^0)$. For simplicity, we first restrict our attention to the planning of the patrol routes at only *current* period without looking into the future periods. We illustrate at the end of this section that how our techniques can be generalized to the planning with look-ahead.

When designing the current patrol routes, the patrolling level l_i^0 at the previous period has already happened, thus is fixed. Therefore, only the input l_i^1 for g_i is under our control. To that end, for notational convenience, we will simply view g_i as a function of l_i^1. In fact, we further omit the superscript "1", and use l_i as the variable of function g_i for simplicity. We start by proving the NP-hardness of the problem. The underlying intuition is that patrolling a cell at different levels will result in different "values" (i.e., the number of detected attacks). Given a fixed budget of patrolling resources, the designer needs to determine which cell has what patrolling level so that it maximizes the total "value". This turns out to encode a Knapsack problem.

Theorem 1. *It is NP-hard to compute the optimal patrolling strategy.*

The proof of Theorem 1 precisely tracks the intuition above, and is omitted here due to space limit.[7] Because of the NP-hardness, the problem of patrolling optimization to maximize attack detection (with inputs: a time-unrolled graph G with $N \times T$ nodes, $\{g_i(j)\}_{i\in[N],j\in[m]}$, $\{\alpha_l\}_{l\in[m]}$) is unlikely to have an efficient polynomial time algorithm. Next, we propose a novel mixed integer linear program (MILP) to solve the problem. We start by observing that the following abstractly described Mathematical Program (MP), with integer variables $\{l_i\}_{i=1}^N$ and real variables $\{x_i\}_{i=1}^N$, encodes the problem.

$$\begin{array}{lll} \text{maximize } \sum_{i=1}^N g_i(l_i) & & \\ \text{subject to } \alpha_{l_i} \le x_i \le \alpha_{l_i+1}, & \text{for } i \in [N]. & \\ \quad l_i \in \{0, 1, ..., m\}, & \text{for } i = 1, ..., N. & \\ \quad (x_1, ..., x_N) \text{ is an implementable effort vector} & & \end{array} \tag{2}$$

We remark that in MP (2), the constraint $\alpha_{l_i} \le x_i < \alpha_{l_i+1}$ is relaxed to $\alpha_{l_i} \le x_i \le \alpha_{l_i+1}$. This is without loss of generality since, in practice, if $x_i = \alpha_{l_i+1}$ for some cell i, we can decrease x_i by a negligible amount of effort and put it anywhere feasible. This will not violate the feasibility constraint but makes x_i strictly less than α_{l_i+1}.

Though MP (2) has complicated terms like $g_i(l_i)$ and α_{l_i} which are *non-linear* in the variable l_i, we show that it can nevertheless be reformulated as a

[7] All missing proofs in this paper can be found in an online appendix.

compactly represented MILP. The main challenge here is to eliminate these non-linear terms. To do so, we introduce m new *binary* variables $\{z_i^j\}_{j=1}^m$, for each i, to encode the integer variable l_i and linearize the objective and constraints of MP (2) using the new variables. By properly constraining the new variables $\{z_i^j\}_{i,j}$, we obtain the following novel MILP (3), which we show is equivalent to MP (2). MILP (3) has binary variables $\{z_i^j\}_{i\in[N],j\in\{1,...,m\}}$ (thus mN binary variables), continuous effort value variables $\{x_i\}_{i\in[N]}$ and flow variables $\{f(e)\}_{e\in E}$. Note however, $g_i(j)$'s are constants given by the black-box attacker model. By conventions, $\sigma^+(v)$ ($\sigma^-(v)$) denotes the set of edges that enter into (exit from) any node v.

$$
\begin{aligned}
&\text{maximize } \sum_{i=1}^N \left(g_i(0) + \sum_{j=1}^m z_i^j \cdot [g_i(j) - g_i(j-1)] \right) \\
&\text{subject to } x_i \geq \sum_{j=1}^m z_i^j \cdot [\alpha_j - \alpha_{j-1}], && \text{for } i = 1, ..., N. \\
&\qquad x_i \leq \alpha_1 + \sum_{j=1}^m z_i^j \cdot [\alpha_{j+1} - \alpha_j], && \text{for } i = 1, ..., N. \\
&\qquad z_i^1 \geq z_i^2 ... \geq z_i^m, && \text{for } i = 1, ..., N. \\
&\qquad z_i^j \in \{0, 1\}, && \text{for } i = 1, ..., N, j = 1, ..., m. \\
&\qquad x_i = \sum_{t=1}^T \left[\sum_{e\in\sigma^+(v_{t,i})} f(e) \right], && \text{for } i = 1, ..., N. \\
&\qquad \sum_{e\in\sigma^+(v_{t,i})} f(e) = \sum_{e\in\sigma^-(v_{t,i})} f(e), && \text{for } i = 1, ..., N; t = 2, ..., T-1. \\
&\qquad \sum_{e\in\sigma^+(v_{T,1})} f(e) = \sum_{e\in\sigma^-(v_{1,1})} f(e) = 1 \\
&\qquad 0 \leq x_i \leq 1, \qquad 0 \leq f(e) \leq 1, && \text{for } i = 1, ..., N; e \in E.
\end{aligned}
\tag{3}
$$

Theorem 2. *MILP* (3) *is equivalent to the Mathematical Program* (2).

Proof. By Lemma 1 and noticing that variable $f(e)$ for all $e \in E$ represents a one-unit flow on graph G, it is easy to verify that the last four sets of constraints in MILP (3) are precisely a mathematical formulation for the constraint "$(x_1, ..., x_N)$ is an implementable effort vector". We therefore only prove that the first four sets of constraints in MILP (3) encode the first two constraints of MP (2). Moreover, the objective functions in MILP (3) and MP (2) are equivalent.

We start by examining the constraints. Since $z_i^1 \geq z_i^2 ... \geq z_i^m$ and $z_i^j \in \{0, 1\}$, we know that any feasible $\{z_i^j\}_{j=1}^m$ corresponds to a $l_i \in \{0, 1, ..., m\}$ such that $z_i^j = 1$ for all $j \leq l_i$ and $z_i^j = 0$ for all $j > l_i$ ($l_i = 0$ means $z_i^j = 0$ for all j). Conversely, given any $l_i \in \{0, 1, ..., m\}$, we can define $z_i^j = 1$ for all $j \leq l_i$ and $z_i^j = 0$ for all $j > l_i$ as a feasible choice of $\{z_i^j\}_{j=1}^m$. That is, there is a one-to-one mapping from feasible $\{z_i^j\}_{j=1}^m$ to feasible l_i for any cell i. Utilizing this one-to-one mapping, we have

$$\sum_{j=1}^m z_i^l \cdot [\alpha_j - \alpha_{j-1}] = \sum_{j=1}^{l_i} [\alpha_j - \alpha_{j-1}] = \alpha_{l_i}.$$

$$\sum_{j=1}^m z_i^j \cdot [\alpha_{j+1} - \alpha_j] + \alpha_1 = \alpha_1 + \sum_{j=1}^{l_i} [\alpha_{j+1} - \alpha_j] = \alpha_{l_i+1}.$$

This shows that any feasible $\{z_i^j\}_{j=1}^m$ encodes an $l_i \in \{0, 1, ..., m\}$ and the first two constraints in MILP (3) are equivalent to $x_i \geq \alpha_{l_i}$ and $x_i \leq \alpha_{l_i+1}$, respectively. The argument for the objective function is similar. In particular, $g_i(0) + \sum_{j=1}^m z_i^j \cdot [g_i(j) - g_i(j-1)] = g_i(0) + \sum_{j=1}^{l_i} [g_i(j) - g_i(j-1)] = g_i(l_i)$. This proves that MILP (3) is equivalent to MP (2), as desired. □

Generalizations. The techniques above can be easily generalized to handle more general tasks and models. First, it works for any defender objective that is linear in g_i's, not necessarily the particular one in MP (2). For example, if the attacks at different cells have different impacts, we can generalize the objective to be a weighted sum of g_i's. Second, the assumption that g_i depends discretely on the patrol level is equivalent to assume that g_i is a piece-wise *constant* function of x_i. This can be further generalized. Particularly, when g_i is a piece-wise *linear* function of x_i, we can still use a similar MILP to solve the problem with a slightly more involved formulation of the objective (e.g., [17]). Finally, when g_i is a continuous function in x_i, its piece-wise linear approximation usually serves as a good estimation of g_i.

Generalization to Route Design with Look-Ahead

We now illustrate how the previous techniques can be generalized to patrol design with look-ahead. The designer will plan for multiple periods and need to take into account the effect of current patrolling on the next period's prediction. For simplicity, we focus on planning for two periods: the current period 1 and the next period 2. The approach presented here can be generalized to planning for any small number of periods. Moreover, in the real-world domain we focus on, there is usually no need for a long-term patrol plan because patrolling resources and environments are dynamic – new plans will need to be frequently designed. The optimal planning for two periods can be formulated as the following mathematical program (MP). Note that here we bring back the omitted superscripts for l_i's to indicate different time periods. Moreover, we use g^1, g^2 to denote the prediction function at period $1, 2$ respectively.

$$
\begin{array}{lll}
\text{maximize } \sum_{i=1}^N g_i^2(l_i^2, l_i^1) + \sum_{i=1}^N g_i^1(l_i^1, l_i^0) & & \\
\text{subject to } \alpha_{l_i^2} \leq x_i^2 \leq \alpha_{l_i^2+1}, & \text{for } i = 1, ..., N. & \\
\quad \alpha_{l_i^1} \leq x_i^1 \leq \alpha_{l_i^1+1}, & \text{for } i = 1, ..., N. & \\
\quad l_i^2 \in \{0, 1, ..., m\}, & \text{for } i = 1, ..., N. & \\
\quad l_i^1 \in \{0, 1, ..., m\}, & \text{for } i = 1, ..., N. & \\
\quad \mathbf{x}^2, \mathbf{x}^1 \text{ are both implementable effort vectors.} &
\end{array}
\tag{4}
$$

MP (4) can also be reformulated as an MILP by employing the techniques above. More precisely, we can introduce binary variables $\{z_i^j\}_{j=1}^m$ and $\{t_i^j\}_{j=1}^m$ to encode l_i^2 and l_i^1 respectively. The additional challenge is to represent $g_i^2(l_i^2, l_i^1)$ as a linear function. To do so, we can equivalently view g_i^2 as a function of $l_i = (m+1)l_i^2 + l_i^1 \in \{0, 1, ..., m^2 + 2m\}$, and introduce $m^2 + 2m$ additional

binary variables $c_i^1, ..., c_i^{m^2+2m}$ for each i, such that $1 \geq c_i^1 \geq ... \geq c_i^{m^2+2m} \geq 0$ and $\sum_{j=1}^{m^2+2m} c_i^j = l_i = (m+1)\sum_{j=1}^{m} z_i^j + \sum_{j=1}^{m} t_i^j$. This guarantees that $c_i^j = 1$ for all $j \leq l_i$ and $c_i^j = 0$ otherwise. Thus $g_i^2(l_i^2, l_i^1) = g_i^2(l_i) = g_i^2(0) + \sum_{j=1}^{m^2+2m} c_i^j \cdot [g_i^2(j) - g_i^2(j-1)]$. So the objective and all the constraints are linear in these variables. Note that this approach introduces $N(m^2+4m)$ binary variables.

5 Increasing Unpredictability via Entropy Maximization

The algorithms in Section 4 output only a flow $\{f_e\}_{e \in E}$ together with the corresponding effort vector. To implement this effort vector in the real world, one needs to decompose the flow to an executable mixed strategy, i.e., a distribution over deterministic patrolling routes. The classical approach is to use a standard flow decomposition algorithm. Unfortunately, these algorithms often output a decomposition with very small number of route choices. For example, in one real-world patrol post we tested, the resulted optimal mixed strategy essentially randomizes over only two patrol routes, as depicted in Figure 2. Despite its optimality, such a mixed strategy is problematic due to its lack of randomness and unpredictability. First, since there are only two routes, the poacher can quickly learn these patrolling routes. Then, knowing these two routes, a poacher can easily figure out where the patroller will be at any time during the day by simply looking at whether their initial move is to the northeast or southwest since this initial move uniquely indicates which route the patroller takes.

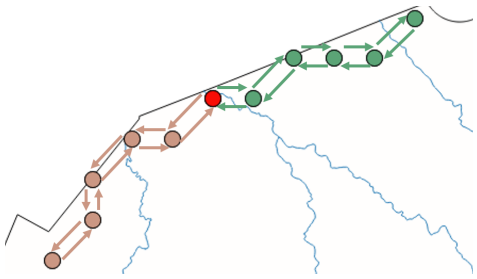

Fig. 2. Visualization of Two Patrol Routes.

To overcome this issue, we seek to compute a mixed strategy that implements the (same) optimal effort vector but is the "most random" in the sense that it has the maximum possible (Shannon) entropy. The underlying intuition is that the increased randomness will make patrolling more unpredictable even when poachers can observe part of the patroller's movement. A thorough experimental justification of the max-entropy approach is done in Section 8.

We start by formulating the problem of computing the mixed strategy that implements the effort vector while maximizing entropy. Let set $\mathcal{P}$ denote the set

of all $s-d$ paths. Our goal is to implement $\{x_i\}_{i\in[N]}$ as a distribution over $\mathcal{P}$ with the maximum entropy, a task which we term the *max-entropy decomposition* of $\{x_i\}_{i\in[N]}$. We will view any $P \in \mathcal{P}$ as a set of nodes in G that specifies the ranger's position at each time step (these nodes uniquely determine the $s-d$ path). Let $P_i = \{v_{t,i} : \exists t, \text{ s.t. } v_{t,i} \in P\}$ denote those nodes corresponding to cell i, so path P patrols cell i with $|P_i|$ units of effort, where $|P_i|$ is the cardinality of P_i. The max-entropy decomposition can be formulated as the following program with variable θ_P representing the probability of picking path P.

$$\begin{array}{ll} \text{maximize} & -\sum_{P\in\mathcal{P}} \theta_P \log \theta_P \\ \text{subject to} & \sum_{P\in\mathcal{P}} |P_i|\theta_P = x_i, \text{ for } i \in E. \\ & \sum_{P\in\mathcal{P}} \theta_P = 1 \\ & \theta_P \geq 0, \qquad\qquad \text{for } P \in \mathcal{P}. \end{array} \tag{5}$$

Observe that program (5) is a convex program (CP) since entropy is a concave function. However, the major challenge of solving CP (5) is that the size of $\mathcal{P}$ (i.e., the total number of $s-d$ paths in G) is exponential in T and therefore so is the total number of variables in CP (5). Indeed, though $T = 12$ in our real-world setting, this results in about 10^{10} variables in CP (5). Such a convex program cannot be efficiently solved by any state-of-the-art optimization software.

To overcome this challenge, we instead examine the *Lagrangian dual* of CP (5) and utilize a well-known characterization of the optimal solution to CP (5) in terms of the optimal solution of its Lagrangian dual – an unconstrained convex program with variables $\{y_i\}_{i\in[N]}$, as follows:

$$\text{Dual of CP (5):} \qquad \min \quad H(\mathbf{y}) = \sum_{i=1}^{N} x_i \cdot y_i + \ln\left[\sum_{P\in\mathcal{P}} \exp(-\sum_{i=1}^{N} |P_i| y_i)\right] \tag{6}$$

Note that $H(\mathbf{y})$ is a convex function. The following well-known lemma characterizes the optimal solution of CP (5) in terms of the optimal solution of CP (6).

Lemma 2 (Adapted from [16]). *Let $\{y_i^*\}_{i\in[N]}$ be the optimal solution to CP (6). Then the optimal solution to CP (5) is given by:*

$$\theta_P^* = \frac{\exp(-\sum_{i=1}^{N} |P_i| y_i^*)}{\sum_{P'\in\mathcal{P}} \exp(-\sum_{i=1}^{N} |P_i'| y_i^*)}, \qquad \forall P \in \mathcal{P} \tag{7}$$

Despite of Lemma 2, two challenges remain in computing the maximum entropy decomposition. The first is to obtain the optimal solution to CP (6). Though CP (6) is a convex program, it is unclear that its objective function can be even evaluated efficiently since $\sum_{P\in\mathcal{P}} \exp(-\sum_{i=1}^{N} |P_i| y_i)$ is a summation of exponentially many terms. To overcome this challenge, we design an efficient dynamic program (DP) to compute the term $\sum_{P\in\mathcal{P}} \exp(-\sum_{i=1}^{N} |P_i| y_i)$ for any given input $\{y_i\}_{i\in[N]}$. The second challenge is that we cannot *explicitly* output the optimal solution $\{\theta_P^*\}_{P\in\mathcal{P}}$ since it takes exponential time to even

write down these many variables. We therefore instead develop a *sampling algorithm* and prove that it samples a path $P \in \mathcal{P}$ with the desired probability θ_P^* in poly(N, T) time. Next, we elaborate our algorithms while omitting formal proofs due to space limit (formal proofs can be found in the online appendix).

For notational convenience, let $C(\mathbf{y})$ denote the term $\sum_{P\in\mathcal{P}} \exp(-\sum_{i=1}^{N} |P_i| y_i)$. To compute $C(\mathbf{y})$, we utilize the natural chronological order along the temporal dimension for nodes in graph G and build a dynamic programming table $DP(t, i)$, for $t \in [T]$ and $i \in [N]$, such that $DP(t, i) = \sum_{P\in\mathcal{P}(t,i)} \exp(-\sum_{i=1}^{N} |P_i| y_i)$ where $\mathcal{P}(t, i)$ is the set of paths from s to the node $v_{t,i}$. We initialize $DP(1, 1) = y_1$ (recall $s = v_{1,1}$) and $DP(1, i) = 0$ for all $i > 1$, and then use the following update rule to return $DP(T, 1)$ (recall $d = v_{T,1}$):

$$DP(t, i) = \sum_{e:e=(v_{t-1,i'}, v_{t,i})} DP(t - 1, i') \cdot \exp(-y_i).$$

Correctness of the algorithm follows a textbook argument. Utilizing this DP, one can efficiently evaluate the objective value of CP (6), and solve the unconstrained optimization problem via any black-box optimization tool (e.g., `fmincon` in MATLAB).

Next, we take $\{y_i^*\}_{i\in[N]}$ as input and efficiently samples an $s - d$ path $P \in \mathcal{P}$ with probability θ_P^*, as defined in Equation (7). The algorithm starts from the node d (=$v_{T,1}$) and at any time t and location loc, samples its predecessor node $v_{t-1,i}$ with probability $p_e = \frac{\exp(-y_{loc}^*)\cdot DP(t-1,i)}{DP(t,loc)}$ where $e = (v_{t-1,i}, v_{t,loc})$. Full details are in Algorithm 1. The following theorem summarizes its correctness.

Theorem 3. *Algorithm 1 correctly samples P with probability θ_P^* for any $P \in \mathcal{P}$ and runs in* poly(N, T) *time, where $\{\theta_P^*\}_{P\in\mathcal{P}}$ is the optimal solution to CP* (5).

Algorithm 1 Max-Entropy Implementation of the Effort Vector

Input: : Effort values at each cell $\{x_i\}_{i\in[N]}$.
Output: : a random path $P \in \mathcal{P}$ which implements $\{x_i\}$ and maximizes entropy.

1: Compute the optimal solution $\{y_i^*\}_{i\in[N]}$ to CP (6) by utilizing the DP.
2: Build the DP table $DP(t, i)$ with $\{y_i^*\}_{i\in[N]}$;
3: Initialize: $P = \{v_{T,1}\}$, Define $loc = 1$;
4: **for** $t = T$ to 2 **do**
5: Sample an edge $e = (v_{t-1,i}, v_{t,loc})$ for all such edges that exist, with probability

$$p_e = \frac{\exp(-y_{loc}^*) \cdot DP(t - 1, i)}{DP(t, loc)};$$

6: Let $e = (v_{t-1,i^*}, v_{t,loc})$ be the sampled edge above, and add v_{t-1,i^*} to P.
7: Update $loc = i^*$.
8: **end for**
9: **return** P.

6 Real-world Dataset

Our analysis focuses on a real-world wildlife crime dataset from Uganda's Queen Elizabeth Protected Area (QEPA). QEPA spans approximately 2,520 square kilometers and is patrolled by wildlife park rangers. While on patrol, they collect data on animal sightings and signs of illegal human activity (e.g., poaching, trespassing). In addition to this observational data, the dataset contains terrain information (e.g., slope, vegetation), distance data (e.g., nearest patrol post), animal density, and the kilometers walked by rangers in an area (i.e., effort).

We divide QEPA into 1 square kilometer grid cells and compute several features based on the dataset's contents (e.g., observations, terrain, effort). Additionally, we group the observations and effort values (i.e., the values that change over time) into a series of month-long time steps. Finally, we compute two effort features, previous effort and current effort, that represent the amount of patrolling effort expended by rangers in the previous time step and current time step, respectively. Because effort is a continuous value (0 to ∞), we discretize the effort values into m effort groups (e.g., $m = 2$: `high` and `low`).

7 Predictive Model Analysis

In this section, we analyze an example attack prediction model that can predict poaching activity for the real-world dataset described in Sect. 6 using an ensemble of decision trees [6]. This model can provide the black-box attack function $g_i(l_i)$ for OPERA and will be used to evaluate OPERA in Sect. 8.

The goal of the analysis is two-fold. First, we analyze the performance of the prediction model to verify that it is a realistic model that can provide $g_i(l_i)$. Second, we analyze how the prediction model's predictions change as a function of ranger effort, which can provide intuition on why planning patrols using OPERA can help increase the efficiency of patrols.

Note that although the hybrid model proposed in [6] is currently the best performing predictive model in this domain, conducting such an analysis on this model may be confounded by its complexity. For instance, the hybrid model's reaction to a change in effort may be due to the underlying bagging ensemble's reaction or it may be due to a reaction in the Markov Random Field that boosts predictions under specific conditions. For scientific rigor, we instead focus on the analysis of a single model's reactivity – the bagging ensemble (which outperforms the previously most accurate model in [9]).

7.1 Ensemble Model

Bagging (Bootstrap aggregation technique) is an ensemble method (in this case applied to decision trees) where each tree is trained on a bootstrapped subset of the whole dataset. The subsets are generated by randomly choosing, with replacement, M observations where M is the dataset size. Once all trees in the ensemble are trained, the ensemble's predictions are generated by averaging the

predictions from each tree. We trained a bagging ensemble using the *fitcensemble* function in MATLAB 2017a. For this model, the best training period consists of 5 years of data (based on repeated analyses for different train/test splits). Described in Sect. 6, the 11 input features consist of terrain and geospatial features, and two patrol effort features (one for previous time step's effort and one for current effort). Each data point's label corresponds to whether an attack was detected at that cell. For the training set, a label will be 1 if at any point in the training period an attack was detected (0 otherwise). For the test set, a label will be 1 if an attack was detected during the current time step.

We present results for a bagging ensemble on a three-month time scale where the ensemble is trained on 5 years of data (effort values are in three-month chunks) and is used to predict detections for a test period of three months. The test set corresponds to September through November 2016, and the training set contains data for 2,129 patrolled cells from September 2012 to August 2016.

In Table 1, we present prediction performance results as verification that subsequent analyses are done on a realistic model. We also present baseline results from common boosting models – AdaBoost and RUSBoost [14]. Additionally, we present a baseline, TrainBaseline, where if an attack was detected at a cell in the training data, the baseline will predict a detected attack for the test data (for cells that were not patrolled in the training data, and thus there is no training sample for that cell, a uniform random binary prediction is made). Due to the large class imbalance present in the dataset (many more negative labels than positives), we compute the area under a Precision-Recall curve (PR-AUC[8]) instead of the standard ROC curve (which is not as informative for such a dataset) [2]. We also present F1, Precision, and Recall scores.

Model	F1	Precision	Recall	PR-AUC
TrainBaseline	0.4	0.25	0.96	-
RUSBoost	0.21	0.12	0.96	0.28
AdaBoost	0.49	0.35	0.82	0.50
Bagging	**0.65**	**0.52**	**0.86**	**0.79**

Table 1. Model Performances

As can be seen, the Bagging model outperforms all other models in terms of F1, Precision, and PR-AUC. While Bagging does not always score the highest in recall, its precision score greatly outperforms the other models' precision. In practical terms, this means that the Bagging model will predict far less false positives and can thus better ensure that the patrol generation algorithm is not needlessly sending rangers to areas where they won't detect attacks.

[8] Because TrainBaseline makes binary predictions and thus does not have continuous prediction values, PR-AUC is not computed for TrainBaseline.

7.2 Effort Function Analysis

The goal of the patrol generation algorithm is to allocate effort such that rangers' detections of attack (poaching activity) are maximized. For the following analysis, we examine how the bagging ensemble's predictions change as a function of ranger effort. For example, if we increase effort in an area over a period of three months, will rangers detect an attack in that area in any of the three months?

For this analysis, we present the changes in (1) the model's detected attack predictions $g_i(l_i)$ and (2) the model's detected attack prediction probabilities when the effort in the current time step is changed. Both values are outputted by MATLAB's *predict* function for our learned ensemble. We refer to effort group 0 as `low` and group 1 as `high`; an increase in allocated effort, for example, would result in l_i changing from `low` to `high`. Results for changes in predictions and prediction probabilities are shown in Tables 2 and 3 respectively.

Effort Change	Neg to Pos	Pos to Neg	No Change (Pos)	No Change (Neg)
Low to High	119	30	172	1693
High to Low	2	110	122	274

Table 2. Prediction Changes as Function of Current Effort

In Table 2, for each type of change in effort (`low` to `high` or `high` to `low`), there are three possible outcomes for a prediction change: a negative prediction (no detection) can change to a positive prediction (detected attack), referred to as **Neg to Pos**, positive can change to negative (**Pos to Neg**), and there can be no change in the prediction (for either the positive or negative prediction cases). Given these outcomes, we make the following observations. First, there are a substantial number of cells whose corresponding detection predictions do not change as a result of changes in effort. In the case of the unchanged positive predictions, these are predicted to be high-risk cells where rangers will find poaching activity even if they allocate relatively low effort values to it. For unchanged negative predictions, these correspond to low-risk cells that are essentially predicted to not be attacked at all. Second, while there are substantially more instances of predicted detections increasing as a result of increasing effort, there are still some instances of predicted detections decreasing as a result of increasing effort. However, because there is not a rational explanation for this trend, these rare instances are likely due to noise in the model. Finally, we make the same observation regarding the case where detections mostly decrease as a result of decreasing effort while detections increase at only two cells.

As for the prediction probability changes in Table 3, we examine changes in the prediction probability with increases and decreases referred to as **Inc** and **Dec** respectively (i.e., any increase or decrease), the mean changes in prediction probability for the increase and decrease cases (referred to as **Mean Inc** and **Mean Dec** respectively), and also in the instances where there was no change in the probability for both the positive (i.e., probability ≥ 0.50) and negative (i.e.,

Effort Change	Inc	Mean Inc	Dec	Mean Dec	No Change(Pos)	No Change(Neg)
Low to High	1546	0.16	423	0.09	4	41
High to Low	142	0.09	358	0.22	0	8

Table 3. Prediction Probability Changes as Function of Current Effort

probability < 0.50) cases. First, when effort is increased, many more cells are predicted to have a substantial increased prediction probability (mean change of 16%). While there are a non-trivial number of cells with a decrease in their prediction probability, the mean decrease is approximately half that of the mean increase, with the difference being statistically significant ($\alpha < 0.01$), and is thus interpreted as noise. Second, when effort is decreased, there are many more cells with a decrease in prediction probability than increase. Additionally, the mean decrease in prediction probability is more than twice that of the mean increase (22% vs 9%) and is also statistically significant ($\alpha < 0.01$). Finally, as with the prediction changes in Table 2, a few cells are low-risk and increasing effort will not result in a corresponding increase in predicted detection probability. While changes in predicted probability do not necessarily correspond to changes in actual predictions (0/1), the shifts in probability do provide a concrete indication of the actual impacts that coverage has on the model's predictions.

8 Experimental Evaluations

8.1 Evaluation of the Patrol Optimization Algorithm

We start by experimentally testing OPERA using the aforementioned real-world data and bagging ensemble predictive model. Particularly, the inputs to all the tested algorithms are specified as follows: graph G is constructed according to the real-world terrain in QEPA; the function g_i's, together with the corresponding $\{\alpha_i\}_{i=1}^m$, are precisely the predictive model described in Section 7 for the test period September through November 2016; $T = 12$ as suggested by domain experts. Since we are not aware of any previous patrol generation algorithm that deals with attackers described by a black-box machine learning model[9], we instead compare our patrol optimization algorithms with the following two heuristic planning algorithms. Note that we will also compare OPERA with its preliminary version without entropy maximization, i.e., the Optimal patrol Planning by flow Decomposition (OPD).

GREED: a heuristic patrol planning algorithm that, at any cell i, greedily picks the next cell j that satisfies: 1. it is feasible to go from i to j; 2. patrolling cell j at `high` is more *beneficial*, (results in more predicted attacks than patrolling j at `low`). If there are multiple such cells, pick one uniformly at random; if there

[9] Most previous algorithms either require knowledge of the patroller's and poacher's payoffs [7, 5] which are not available in our setting or generates patrolling strategies that are not guaranteed to be implementable [11].

are no such cells, then pick any neighbor cell uniformly at random. To guarantee that the patrol path starts and ends at the patrol post, this procedure continues until time $\lceil T/2 \rceil$ and then the patroller returns via its outgoing route.

RAND: a heuristic patrol planning algorithm that is similar to GREED except that at any cell i, it chooses a neighbor cell j to go uniformly at random without considering the prediction model.

There are 39 patrol posts at QEPA. We test the algorithms on the real data/model at patrol post 11, 19 and 24, which are the three posts that have the most attacks in the three months of our testing. In our data, all posts have less than 100 cells/targets reachable from the post by a route of maximum duration T=12 (equivalently, a 12-cells long route) and all the algorithms scale very well to this size (the MILP takes at most 2 seconds in any tested instance). We thus focus on comparing these algorithms' ability in detecting attacks under multiple criteria, as follows:

- **#Detection**: total number of detected attacks under the prediction model. Since the prediction model we adopt is a 0-1 classification algorithm, in this case **#Detection** also equals the number of cells at which the corresponding patrolling levels result in detected attacks. However, here we exclude those cells for which `high` or `low` patrol effort results in the same prediction because patrolling levels at these cells do not make a difference to the criterion.
- **#Cover**: total number of cells that are patrolled with `high`. Note that due to limited resources, not every cell – in fact, only a small fraction of the cells – can be covered with `high`.
- **#Routes**: the number of different patrol routes in 90-day route samples (corresponding to a 3-month patrolling period).
- **Entropy**: The entropy of the empirical distribution of the 90 samples.

Note that the last two criteria are used particularly to test the unpredictability of these algorithms in an environment with partial observations by the attacker. A higher value of **#Routes** means that the patroller has more choices of patrol routes, thus less explorable by the poacher. **Entropy** is a natural measure to quantify uncertainty. The experimental results for patrol post 11 and 19 are shown in Table 4 and 5, respectively. The results for post 24 are similar to that for post 19, thus are omitted here to avoid repetition. For the **#Detection** criterion, a/b means that out of the b cells for which `low` or `high` makes a prediction difference, a of them are "hit" correctly – i.e., patrolled at the right level that results in predicted attack detection – by the patrolling algorithm. For example, in Table 4, the "15/19" comes from the follows: there are 19 cells at the post for which patrol level `high` or `low` makes a difference in attack detection; The patrol levels by OPERA result in positive attack detections in 15 out of these 19 cells. For the **#Cover** criterion, a/b means that out of b cells in total, a cells are patrolled with `high`. From the analysis in Section 7 we know that compared to the `low` patrol level, the `high` patrol level is more likely to, though not always, result in attack detections. Therefore, a larger **#Cover** value will be preferred in our comparisons.

	#Detection	#Cover	#Routes	Entropy
OPERA	15/19	20/47	61	4.0
OPD	15/19	20/47	10	2.0
GREED	5/19	4/47	84	4.4
RAND	4/19	6/47	89	4.5

Table 4. Comparisons of Different Criteria for Patrol Post 11

	#Detection	#Cover	#Routes	Entropy
OPERA	6/6	24/72	22	2.6
OPD	6/6	24/72	6	1.3
GREED	2/6	2/72	1	0
RAND	2/6	6/72	90	4.5

Table 5. Comparisons of Different Criteria for Patrol Post 19

As we can see from both tables, OPERA and OPD[10] result in significantly more detected attacks and cells with `high` coverage than the GREED and RAND heuristics. GREED results in slightly more detected attacks than RAND, but RAND covers more cells with `high`. This is because GREED biases towards cells that need more patrolling, thus easily gets concentrated on these cells. For the **#Routes** and **Entropy** criteria, RAND is the most unpredictable (as expected). GREED is unstable. Particularly, at patrol post 19, GREED always chooses the same path. This is because it reaches a cell for which `high` is better and gets stuck at the same cell always due to its greedy choice. This is a critical drawback of GREED. In fact, the same phenomenon is also observed at post 24. Clearly, OPERA exhibits more unpredictability than OPD, and is more stable than GREED. This shows that among these tested algorithms, OPERA provides the best balance among unpredictability, stability and the ability in detecting attacks and covering more cells.

8.2 Comparisons With the Past Patrol Routes

We now compare the patrol routes generated by OPERA with the past patrol routes used by rangers at QEPA. We still adopt the measures in the above Section 8.1. Since there is no ground truth to compare with (for the past patrolling, we do not know what happened at those cells that are not patrolled), as an approximation we will treat the bagging ensemble predictive model described in Section 7 as the ground truth. This is a reasonable choice since [6] recently

[10] Note: they always have the same **#Detection** and **#Cover** since they are both optimal.

shows that this model outperforms all previous poaching prediction models and provides relatively accurate predictions on the QEPA dataset.

Criteria	Post 11		Post 19		Post 24	
	OPERA	Past	OPERA	Past	OPERA	Past
#Detections	15/19	4/19	6/6	5/6	4/4	3/4
#Cover	20/47	6/47	24/72	11/72	20/59	14/59
#Routes	61	4	22	33	34	5
#Entropy	4.0	1.2	2.6	3	2.8	1.4

Table 6. Comparisons of Different Criteria at different Patrol Posts

The results are jointly presented in Table 6. As we can see, the patrol routes generated by OPERA clearly outperform past patrolling in terms of the **#Detections** and **#Cover** criteria. Particularly, the routes we generate can detect attacks on most (if not all) cells by properly choosing their patrolling levels and also result in more cells covered with `high`. In terms of unpredictability, the past patrolling does not have a stable performance. Particularly, it follows only a few routes at post 11 and 24 with low unpredictability but takes many different routes at post 19 with high unpredictability. This is a consequence of various factors at different posts, like patroller's preferences, location of the patrol post (e.g., inside or at the boundary of the area), terrain features, etc. On the other hand, OPERA always comes with good unpredictability guarantee. This shows the advantage of OPERA over the past patrolling.

9 Conclusion

In this paper, we presented a general patrol planning framework OPERA. It can optimize against a wide range of prediction models and generate implementable patrol strategies. In addition, OPERA maximizes the randomness of generated strategies and increases robustness against partial information leakage (i.e., poachers may infer the patroller's patrolling route by monitoring part of the patroller's movements). Experimental results on a real-world data set from Uganda's Queen Elizabeth Protected Area (QEPA) show that OPERA results in better defender strategies than heuristic planning algorithms and the past real patrol routes used by rangers at QEPA in terms of defender utility, coverage of the area and unpredictability.

Acknowledgements. Part of this research is supported by NSF grant CCF-1522054. Fei Fang is partially supported by the Harvard Center for Research on Computation and Society fellowship.

Bibliography

[1] Critchlow, R., Plumptre, A.J., Alidria, B., Nsubuga, M., Driciru, M., Rwetsiba, A., Wanyama, F., Beale, C.M.: Improving law-enforcement effectiveness and efficiency in protected areas using ranger-collected monitoring data. Conservation Letters (2016)

[2] Davis, J., Goadrich, M.: The relationship between precision-recall and roc curves. In: Proceedings of the 23rd International Conference on Machine Learning. ICML (2006)

[3] Di Marco, M., Boitani, L., Mallon, D., Hoffmann, M., Iacucci, A., Meijaard, E., Visconti, P., Schipper, J., Rondinini, C.: A retrospective evaluation of the global decline of carnivores and ungulates. Conservation Biology 28(4), 1109–1118 (2014)

[4] Fang, F., Nguyen, T.H., Pickles, R., Lam, W.Y., Clements, G.R., An, B., Singh, A., Tambe, M., Lemieux, A.: Deploying paws: Field optimization of the protection assistant for wildlife security. In: Twenty-Eighth IAAI Conference (2016)

[5] Fang, F., Stone, P., Tambe, M.: When security games go green: Designing defender strategies to prevent poaching and illegal fishing. In: Twenty-Fourth International Joint Conference on Artificial Intelligence (2015)

[6] Gholami, S., Ford, B., Fang, F., Plumptre, A., Tambe, M., Driciru, M., Wanyama, F., Rwetsiba, A., Nsubaga, M., Mabonga, J.: Taking it for a test drive: A hybrid spatio-temporal model for wildlife poaching prediction evaluated through a controlled field test. In: Proceedings of the European Conference on Machine Learning & Principles and Practice of Knowledge Discovery in Databases. ECML PKDD 2017 (2017)

[7] Haas, T.C., Ferreira, S.M.: Optimal patrol routes: interdicting and pursuing rhino poachers. Police Practice and Research pp. 1–22 (2017)

[8] Haghtalab, N., Fang, F., Nguyen, T.H., Sinha, A., Procaccia, A.D., Tambe, M.: Three strategies to success: learning adversary models in security games. In: Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence. pp. 308–314. AAAI Press (2016)

[9] Kar, D., Ford, B., Gholami, S., Fang, F., Plumptre, A., Tambe, M., Driciru, M., Wanyama, F., Rwetsiba, A., Nsubaga, M., Mabonga, J.: Cloudy with a chance of poaching: Adversary behavior modeling and forecasting with real-world poaching data. In: Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems. pp. 159–167. AAMAS '17 (2017)

[10] Moreto, W.: To conserve and protect: Examining law enforcement ranger culture and operations in Queen Elizabeth National Park, Uganda. Ph.D. thesis, Rutgers University-Graduate School-Newark (2013)

[11] Nguyen, T.H., Sinha, A., Gholami, S., Plumptre, A., Joppa, L., Tambe, M., Driciru, M., Wanyama, F., Rwetsiba, A., Critchlow, R., et al.: Capture: A new predictive anti-poaching tool for wildlife protection. In: Proceedings

of the 2016 International Conference on Autonomous Agents & Multiagent Systems. pp. 767–775. International Foundation for Autonomous Agents and Multiagent Systems (2016)
[12] Nguyen, T.H., Yang, R., Azaria, A., Kraus, S., Tambe, M.: Analyzing the effectiveness of adversary modeling in security games. In: AAAI (2013)
[13] Nyirenda, V.R., Chomba, C.: Field foot patrol effectiveness in kafue national park, zambia. Journal of Ecology and the Natural Environment 4(6), 163–172 (2012)
[14] Seiffert, C., Khoshgoftaar, T.M., Van Hulse, J., Napolitano, A.: Rusboost: A hybrid approach to alleviating class imbalance. IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans 40(1), 185–197 (2010)
[15] Shieh, E., An, B., Yang, R., Tambe, M., Baldwin, C., DiRenzo, J., Maule, B., Meyer, G.: Protect: A deployed game theoretic system to protect the ports of the united states. In: Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems-Volume 1. pp. 13–20. International Foundation for Autonomous Agents and Multiagent Systems (2012)
[16] Singh, M., Vishnoi, N.K.: Entropy, optimization and counting. In: STOC. pp. 50–59. ACM (2014)
[17] Wolsey, L.A.: Integer programming. Wiley-Interscience, New York, NY, USA (1998)
[18] Xu, H., Tambe, M., Dughmi, S., Noronha, V.L.: The curse of correlation in security games and principle of max-entropy. CoRR abs/1703.03912 (2017)
[19] Yang, R., Ford, B., Tambe, M., Lemieux, A.: Adaptive resource allocation for wildlife protection against illegal poachers. In: Proceedings of the 2014 international conference on Autonomous agents and multi-agent systems. pp. 453–460. International Foundation for Autonomous Agents and Multiagent Systems (2014)
[20] Yin, Z., Jiang, A.X., Tambe, M., Kiekintveld, C., Leyton-Brown, K., Sandholm, T., Sullivan, J.P.: Trusts: Scheduling randomized patrols for fare inspection in transit systems using game theory. AI Magazine 33(4), 59 (2012)

Online Appendix

Proof of Theorem 1

We prove by reducing the *Knapsack problem* to the design of the optimal patrolling strategy. A Knapsack instance is described as follows: there are m items; item i has weight $w_i \in (0,1)$ and value v_i. With a knapsack of weight capacity 1, the goal is to pick items to maximize the total value subject to the knapsack's capacity. We assume, w.l.o.g., $w_1 < w_2 .. < w_m$. We reduce this problem to the computation of the optimal patrolling strategy for the following constructed instance. There are three time steps ($T = 3$) and $N = m + 1$ cells. For notational convenience, and only in this proof, let cell $m + 1$ be the patrol post; $v_{1,m+1}$ connects to $v_{2,i}$ which then connects to $v_{3,m+1}$ for any $i = 1, ..., m$. That is, the patrol post can reach any cell $i \in \{1, ..., m\}$ which then reaches the patrol post. Each cell can be patrolled at level $1, ..., m$. Define the effort threshold $\alpha_l = w_l$ for $l = 1, ..., m$. For any cell $i = 1, ..., m$, define $g_i(l) = 0$ for all $l \neq i$ and $g_i(i) = v_i$. Let $g_{m+1}(l) == 0$ for any $l = 0, 1, .., m$. That is, patrolling the patrol post never results in any detection of attacks.

By construction, the defender can detect attacks at cell i only when it is patrolled at level $l = i$, which results in detection of $g_i(i) = v_i$ attacks and takes patrolling effort $x_i \in [\alpha_i, \alpha_{i+1})$ where $\alpha_i = w_i$ and $\alpha_{i+1} = w_{i+1}$. Note that, in the constructed instance above, there is no benefit to patrol a cell more than w_i if the targeted patrolling level is i. As a result, to maximize the total number of detected attacks, the defender needs to select a subset of cells S so that any cell $i \in S$ is patrolled with level i (i.e., has patrolling effort w_i), the total patrolling effort is upper bounded by 1, and the total number of detected attacks $\sum_{i \in S} v_i$ is maximized. This is precisely the Knapsack problem.

Proof of Theorem 3

It is easy to see that each step of Algorithm 1 runs in poly(N, T) time. We only mention that in Step 1, evaluating the function objective in polynomial time via the dynamic program allows us to solve the optimization problem in polynomial time. Moreover, for any t, loc, we have

$$\sum_{e:e=(v_{t-1,i},v_{t,loc})} p_e = \sum_{e:e=(v_{t-1,i},v_{t,loc})} \frac{exp(-y^*_{loc}) \cdot DP(t-1,i)}{DP(t,loc)} = 1,$$

since $DP(t, loc) = \sum_{e:e=(v_{t-1,i},v_{t,loc})} exp(-y^*_{loc}) \cdot DP(t-1, i)$ according to the DP updating rule. Therefore, p_e's for all edges that reach $v_{t,loc}$ do form a probability distribution. We now show that each path P is sampled with the desired probability θ^*_P. Observe that any sampled path will be from $s = v_{1,1}$ to $d = v_{T,1}$. This is because $D(1, i) = 0$ for all $i \neq 1$ and thus only $v_{1,1}$ will be in P. Moreover, the node $v_{T,1}$ is always in P according to the definition of Algorithm1 (specifically, Step 3).

Let $P = \{v_{1,1}, v_{2,i_2}, ...v_{T-1,i_{T-1}}, v_{T,1}\}$ be any sampled $s-d$ path. For notational convenience, we use e_t to denote the edge $(v_{t-1,i_{t-1}}, v_{t,i_t})$. We have

$$\begin{aligned}
\mathbf{Pr}(P) &= \prod_{t=2}^{T} \mathbf{Pr}[\text{sample the edge } e_t] \\
&= \prod_{t=2}^{T} \frac{\exp(-y^*_{i_t}) \cdot DP(t-1, i_{t-1})}{DP(t, i_t)} \\
&= \left[\prod_{t=2}^{T} \exp(-y^*_{i_t})\right] \times \left[\prod_{t=2}^{T} \frac{DP(t-1, i_{t-1})}{DP(t, i_t)}\right] \\
&= \prod_{t=2}^{T} \exp(-y^*_{i_t}) \cdot \frac{D(1,1)}{D(T,1)} \\
&= \prod_{t=2}^{T} \exp(-y^*_{i_t}) \cdot \frac{\exp(-y^*_1)}{D(T,1)} = \theta^*_P.
\end{aligned}$$

This concludes the proof.