Improving Community-Participated Patrol for Anti-Poaching

Yufei Wu[1], Yixuan Even Xu[2], Xuming Zhang[3], Duo Liu[3], Shibing Zhu[4], Fei Fang[2]
[1]Shanghai Jiao Tong University
[2]Carnegie Mellon University
[3]World Wide Fund For Nature China
[4]Heilongjiang Academy of Sciences
yvonne2021@sjtu.edu.cn, yixuanx@cs.cmu.edu, uriel900401@gmail.com,
dliu@wwfchina.org, zhushibing@haszrs.org.cn, feif@cs.cmu.edu

Abstract

Community engagement plays a critical role in anti-poaching efforts, yet existing mathematical models aimed at enhancing this engagement often overlook direct participation by community members as alternative patrollers. Unlike professional rangers, community members typically lack flexibility and experience, resulting in new challenges in optimizing patrol resource allocation. To address this gap, we propose a novel game-theoretic model for community-participated patrol, where a conservation agency strategically deploys both professional rangers and community members to safeguard wildlife against a best-responding poacher. In addition to a mixed-integer linear program formulation, we introduce a Two-Dimensional Binary Search algorithm and a novel Hybrid Waterfilling algorithm to efficiently solve the game in polynomial time. Through extensive experiments and a detailed case study focused on a protected tiger habitat in Northeast China, we demonstrate the effectiveness of our algorithms and the practical applicability of our model.

Code — https://github.com/YvonneWu10/Community-Participated-Patrol

Extended version — http://arxiv.org/abs/2412.10799

1 Introduction

Community engagement has become increasingly recognized as a vital element in the fight against wildlife poaching (Gill et al. 2014; Nubani et al. 2023; Wilson-Holt and Roe 2021). By involving local communities directly in conservation efforts, agencies can leverage local knowledge and foster a sense of ownership over the protection of natural resources. Among various strategies, community-participated patrols, where community members actively engage in monitoring and protecting wildlife, have gained popularity as an effective means of enhancing anti-poaching initiatives (Massé et al. 2017; Danoff-Burg and Ocañas 2022). These patrols not only supplement the efforts of professional rangers but also strengthen the ties between conservation agencies and the communities they serve.

Despite the rise of community-participated patrols, existing game-theoretic models for anti-poaching resource allocation have largely overlooked the direct role of community members as patrollers. Most models focus on indirect engagement, like community reporting poacher locations to rangers (Sjöstedt et al. 2022; Linkie et al. 2015; Huang et al. 2020; Shen et al. 2020, 2024), without addressing the complexities of deploying community members in the field. Differences in flexibility and experience between community members and professional rangers pose unique challenges in optimizing patrols, which remain underexplored. For example, in Northeast China, each community member is assigned to a 2km by 2km area and will patrol the same area at least twice a week throughout the patrol season. In contrast, professional rangers cover multiple areas every week, with the choices of areas changing weekly.

To address this gap, we propose a novel game-theoretic model that explicitly incorporates community-participated patrols into the strategic allocation of anti-poaching resources. Our model enables conservation agencies to assign both professional rangers and community members (or villagers) to multiple target areas, accounting for the poacher's choice of the target with the highest expected utility after observing the patrol strategy. Rangers follow a mixed strategy, with assignments based on a probability distribution over targets. The randomness in this strategy makes it harder for poachers to avoid detection. In contrast, villagers, due to limited flexibility, follow a deterministic strategy, with each assigned to a specific target.

To solve the game, we first present a mathematical program-based solution. The deterministic allocation for villagers introduces integer variables, resulting in an NP-hard mixed-integer linear program. We then design two polynomial-time algorithms that leverage key properties of the problem. The first, the Two-Dimensional Binary Search algorithm, is an approximation method that performs binary searches on both the number of villagers and the number of rangers assigned to the poacher's chosen target. The second, the Hybrid Waterfilling algorithm, provides an exact solution by combining the "water-filling" idea (Kiekintveld et al. 2009) from security games with binary search and iterative target swapping for villagers. We validate our approach's effectiveness and computational efficiency through synthetic data experiments. We further run an extensive case study on a protected tiger habitat in Northeast China with real-world data, demonstrating the potential of our model in real-world conservation scenarios. We have presented the model to conservation agencies in 13 countries and we plan to work with the local forest bureau in Northeast China to adjust future patrol resource allocation based on the case study results.

2 Related Work

Many works on patrol allocation in security and environmental sustainability domains consider homogeneous patrol resources (Fang, Stone, and Tambe 2015; Johnson, Fang, and Tambe 2012). For heterogeneous patrol resources, there have been several works studying the synergy between rangers and mobile sensors for anti-poaching (Basilico et al. 2017; Xu et al. 2018; Bondi et al. 2020). The sensors can be used for detecting poaching and informing patrollers, but can not stop the poachers themselves. The sensors can also signal the poachers that rangers are coming to deter them from poaching. Therefore, these works pay attention to the design of the joint allocation of rangers and sensors to raise the probability that the rangers come when poaching is detected and the strategic signaling scheme of the mobile sensors to better deter poachers. Unlike sensors, villagers can stop poaching by removing the snares but can only follow deterministic allocation. Therefore, the main technical focus of our work is the combination of deterministic allocation of villagers and random allocation of rangers.

In addition, Mc Carthy et al. (2016) proposes a model for the optimal selection and deployment of security resource teams with varying effectiveness and costs. The model is extended in McCARTHY et al. (2018), which considers the inflexibility of resources that defenders are only allowed to be distributed to a set of targets in given periods, resulting in an NP-Hard problem. In contrast, we specify the inflexibility that villagers can not be allocated to different targets and provide a polynomial-time algorithm.

Coordinating multiple defenders is widely studied in security domains (Mutzari, Gan, and Kraus 2021; Castiglioni, Marchesi, and Gatti 2021; Gutierrez et al. 2023). These works usually consider that different defenders have different objectives and focus on the design of stable agreements among defenders. In our case, we view the rangers and villagers as being fully cooperative.

The idea of "water-filling" has been used in the design of algorithms for solving mathematical models for security problems (Kiekintveld et al. 2009; Nguyen et al. 2015; Nguyen and Xu 2019; Gan et al. 2019). By treating each target as a bucket and patrol resources as water, the algorithm fills water to the bucket with the minimum water level. A key factor in using the water-filling algorithm is that the patrol resources can be divided and allocated to multiple targets at will so that they can be treated as water. However, the villagers in our problem can not be distributed to different targets, which breaks this property. Extending the algorithm to handle the villagers is a highly non-trivial task, and we design our Hybrid Waterfilling Algorithm by combining water filling with binary search and iterative swapping to address the challenge as detailed in later sections.

3 Problem Formulation

We propose and study the *Resources Allocation of Community Participated Patrol* (RACPP) problem. This problem formulation builds upon the Stackleberg Security Game (SSG) formulation (Tambe 2011) where a defender allocates patrol resources to protect a set of targets against a best-responding attacker. However, the key difference in RACPP is that two types of patrol resources co-exist: professional rangers and community members who are less flexible and effective. In the anti-poaching domain, community members are from nearby villages and towns who are less experienced in finding poaching tools during patrols and are often only willing to go to a fixed area (i.e., one target) for patrols throughout the patrol season as it is easier for them to plan their other daily work. For expository purposes, we will refer to them as villagers in the rest of the paper.

More concretely, a defender can allocate a group of r^{p} rangers and r^{v} villagers to protect n targets $T = \{0, 1, \dots, n-1\}$. A ranger can distribute their[1] efforts among multiple targets, while a villager can only be allocated to a single target. We denote the defenders' defensive strategy profile as a tuple $(\mathbf{p}, \mathbf{v})$, where $\mathbf{p} = (p_0, p_1, \dots, p_{n-1})$ and $\mathbf{v} = (v_0, v_1, \dots, v_{n-1})$. Here, $p_i \in \mathbb{R}_{\geq 0}$ is the amount of rangers' effort distributed to target i and $v_i \in \mathbb{N}$ is the number of villagers patrolling target i. A *valid* defender strategy profile must satisfy

$$\sum_{i \in T} p_i \leq r^{\mathrm{p}}, \quad \sum_{i \in T} v_i \leq r^{\mathrm{v}}. \tag{1}$$

The attacker will select a target $i \in T$ to attack to maximize their expected utility after observing the defender strategy. The probability of successfully attacking the target depends on the *coverage* of that target provided by the defender. Specifically, let the defense effectiveness of one unit of ranger and one villager be e^{p} and e^{v}, respectively. The total coverage on target i is

$$c_i = \min(e^{\mathrm{p}} \cdot p_i + e^{\mathrm{v}} \cdot v_i, 1). \tag{2}$$

We use the coverage vector $\mathbf{c} = (c_0, c_1, \dots, c_{n-1})$ to denote the coverage on all targets. If target i is attacked, the probability of successfully defending the target is c_i. The attacker and defender receive rewards and penalties based on the outcome of the attack. If target i is successfully defended, the defender receives reward R_i^{d} and the attacker receives penalty P_i^{a}. Otherwise, the defender receives penalty P_i^{d} and the attacker receives reward R_i^{a}. Here, we assume $R_i^{\mathrm{d}} \geq 0 \geq P_i^{\mathrm{d}}$ and $R_i^{\mathrm{a}} \geq 0 \geq P_i^{\mathrm{a}}$. Then, the expected utility of defenders and the attacker are respectively

$$\begin{aligned} U_i^{\mathrm{d}} &= R_i^{\mathrm{d}} \cdot c_i + P_i^{\mathrm{d}} \cdot (1 - c_i), \\ U_i^{\mathrm{a}} &= R_i^{\mathrm{a}} \cdot (1 - c_i) + P_i^{\mathrm{a}} \cdot c_i. \end{aligned} \tag{3}$$

The attacker always chooses to attack the target that maximizes their expected utility. If there are multiple targets that maximize the attacker's expected utility, the attacker will choose the one that maximizes the defender's expected utility following the standard SSG model (Tambe 2011). Thus, given the defender strategy profile $(\mathbf{p}, \mathbf{v})$, the attacker's response is fixed. We can then define the defenders' expected utility as a function of the defender strategy profile, $u(\mathbf{p}, \mathbf{v})$. The defenders' goal is to maximize their expected utility by adjusting their defensive strategy $(\mathbf{p}, \mathbf{v})$ as the attacker selects target i to attack with *best response*.

[1] We use their instead of his or her in this paper.

Definition 3.1. *An **input instance** of the RACPP is a tuple* $\mathcal{I} = (n, r^{\mathrm{p}}, r^{\mathrm{v}}, e^{\mathrm{p}}, e^{\mathrm{v}}, \mathbf{R}^{\mathrm{d}}, \mathbf{P}^{\mathrm{d}}, \mathbf{R}^{\mathrm{a}}, \mathbf{P}^{\mathrm{a}})$.

4 Algorithms for RACPP

In this section, we will present our algorithms to solve the RACPP problem. First, in Section 4.1, we show that the RACPP problem can be formulated as a mixed-integer linear program (MILP). This MILP is easy to understand, but its runtime is exponential in the worst case. We then show in Section 4.2 our Two-Dimensional Binary Search algorithm which solves the RACPP problem to any given accuracy ε in $O(n^2 \log \frac{M}{\varepsilon})$, where M is the maximum absolute value of the reward and penalties. Finally, in Section 4.3, we present an exact algorithm named Hybrid Waterfilling algorithm that solves RACPP precisely in $O(n^4 \log n)$.

4.1 Mixed-Integer Linear Program Solution

The RACPP problem is a Stackelberg game (Stackelberg 1934), where the defender is the leader and the attacker is the follower. Suppose we already know that target i^* is the target to be attacked in equilibrium. To ensure target i^* is the attacker's best response, we need $U^{\mathrm{a}}_{i^*} \geq U^{\mathrm{a}}_i, \forall i \in T$ (Conitzer and Sandholm 2006), and we would like to maximize the defender utility $U^{\mathrm{d}}_{i^*}$ subjected to (1), (2) and (3). If we consider the defender strategy profile $(\mathbf{p}, \mathbf{v})$ as a set of variables, then, the RACPP problem can be formulated as an optimization problem. This problem can be converted into a MILP by using common techniques from the MILP literature (Bradley, Hax, and Magnanti 1977) to linearize (2). To enforce c_i to be $\min(e^{\mathrm{p}} \cdot p_i + e^{\mathrm{v}} \cdot v_i, 1)$, we introduce a large number M (dependent on the instance), continuous variables $\boldsymbol{\delta}$ and binary variables $\mathbf{w}$. For each $i \in T$, we require $w_i \leq c_i \leq 1, \delta_i \leq M \cdot w_i$ and $c_i + \delta_i = e^{\mathrm{p}} \cdot p_i + e^{\mathrm{v}} \cdot v_i$. When $e^{\mathrm{p}} \cdot p_i + e^{\mathrm{v}} \cdot v_i \leq 1$, w_i and δ_i are limited to 0, thus $c_i = e^{\mathrm{p}} \cdot p_i + e^{\mathrm{v}} \cdot v_i$; Otherwise, c_i is set to 1 since w_i is limited to 1. The complete MILP is as follows.

$$
\begin{array}{r|lll}
\textbf{Maximize} & U^{\mathrm{d}}_{i^*} & & \\
\textbf{Subject to} & U^{\mathrm{a}}_{i^*} \geq U^{\mathrm{a}}_i & & (\forall i \in T) \\
 & c_i + \delta_i = e^{\mathrm{p}} \cdot p_i + e^{\mathrm{v}} \cdot v_i & & (\forall i \in T) \\
 & \delta_i \leq M \cdot w_i, & w_i \leq c_i \leq 1 & (\forall i \in T) \\
 & U^{\mathrm{a}}_i = R^{\mathrm{a}}_i \cdot (1 - c_i) + P^{\mathrm{a}}_i \cdot c_i & & (\forall i \in T) \\
 & U^{\mathrm{d}}_i = R^{\mathrm{d}}_i \cdot c_i + P^{\mathrm{d}}_i \cdot (1 - c_i) & & (\forall i \in T) \\
 & p_i \geq 0, & v_i \in \mathbb{N} & (\forall i \in T) \\
 & \delta_i \geq 0, & w_i \in \{0, 1\} & (\forall i \in T) \\
 & \sum_{i \in T} p_i \leq r^{\mathrm{p}}, & \sum_{i \in T} v_i \leq r^{\mathrm{v}} &
\end{array}
$$

By enumerating all targets as the attacked target i^*, we can solve the RACPP problem by solving the MILP for each target. However, the runtime of the MILP solution is exponential in the worst case since MILP is NP-hard (Karp 2010). In the rest of the paper, we aim to design a polynomial-time algorithm to solve the RACPP problem.

4.2 A Polynomial Approximate Algorithm: Two-Dimensional Binary Search

In this section, we will present a polynomial-time algorithm that solves the RACPP problem to any desired accuracy ε. The algorithm is based on a two-dimensional binary search.

To begin with, we first consider the following decision problem: given a target i^* and a fixed defender strategy (p_{i^*}, v_{i^*}) on i^*, can we find a strategy profile $(\mathbf{p}, \mathbf{v})$ such that the attacker will choose to attack the target i^* with best response? We can solve this problem by greedily distributing the remaining resources to other targets. The following Algorithm 1 checks whether a consistent strategy exists.

Algorithm 1: Checking whether a consistent strategy exists

Input: Input instance $\mathcal{I}$, target i^*, strategy (p_{i^*}, v_{i^*}) on i^*
Output: Whether a consistent strategy $(\mathbf{p}, \mathbf{v})$ exists

1: Compute $U^{\mathrm{a}}_{i^*}$ using (2), (3) with (p_{i^*}, v_{i^*}).
2: **if** $\exists i \in T, U^{\mathrm{a}}_{i^*} < P^{\mathrm{a}}_i$ **then**
3:     **return** False.
4: Let $(r^{\mathrm{p}}_{\mathrm{remain}}, r^{\mathrm{v}}_{\mathrm{remain}}) \leftarrow (r^{\mathrm{p}} - p_{i^*}, r^{\mathrm{v}} - v_{i^*})$.
5: Let $\delta_i \leftarrow 0$ **for all** $i \in T$.
6: **for** $i \in \{0, 1, \ldots, n-1\} \setminus \{i^*\}$ **do**
7:     Let $c_{\min,i} \leftarrow$ the minimum c_i to ensure $U^{\mathrm{a}}_i \leq U^{\mathrm{a}}_{i^*}$.
8:     Let $v_{\mathrm{cnt},i} \leftarrow \min\left(\lfloor c_{\min,i}/e^{\mathrm{v}} \rfloor, r^{\mathrm{v}}_{\mathrm{remain}}\right)$.
9:     Let $r^{\mathrm{v}}_{\mathrm{remain}} \leftarrow r^{\mathrm{v}}_{\mathrm{remain}} - v_{\mathrm{cnt},i}$.
10:     Let $\delta_i \leftarrow c_{\min,i} - v_{\mathrm{cnt},i} \cdot e^{\mathrm{v}}$.
11: **for** $i \in \{0, 1, \ldots, \min(n, r^{\mathrm{v}}_{\mathrm{remain}}) - 1\}$ **do**
12:     Let the largest δ_j $(j \in T) \leftarrow 0$.
13: **return** $\sum_{i \neq i^*} \delta_i \leq r^{\mathrm{p}}_{\mathrm{remain}} \cdot e^{\mathrm{p}}$.

Definition 4.1. *Given an utility u, the **minimum valid coverage** is a vector $\mathbf{c}_{\min}(u)$, where $c_{\min,i}(u)$ is the minimum coverage on target i such that $U^{\mathrm{a}}_i \leq u$.*

Definition 4.2. *Given a villager strategy $\mathbf{v}$ and an utility u, the **wasted villager coverage** is a vector $\mathbf{c}^{\mathrm{w}}(\mathbf{v}, u)$, where $c^{\mathrm{w}}_i(\mathbf{v}, u) = \max(v_i \cdot e^{\mathrm{v}} - c_{\min,i}(u), 0)$, and the total wasted villager coverage is $\mathrm{scw}(\mathbf{v}, u) = \sum_{i \neq i^*} c^{\mathrm{w}}_i(\mathbf{v}, u)$.*

Algorithm 1 first checks whether there are targets that allow the attacker utility to be higher than $U^{\mathrm{a}}_{i^*}$ no matter how many resources are allocated to them (Lines 2). If there is such a target, then it is impossible to make i^* the attacker's best response (Line 3). Otherwise, the algorithm will try to allocate resources to other targets to ensure that the attacker's utility is no more than $U^{\mathrm{a}}_{i^*}$ (Lines 4 to 12). To do this, we first calculate the minimum coverage $c_{\min,i}$ for each target i (Line 7). The problem is then to check if it is feasible to achieve the required minimum coverage on each target.

We allocate the resources greedily: we try first to allocate villagers and then rangers. Intuitively, this is because villagers can only be allocated to one target as a whole, while rangers are more flexible in distributing their effort to multiple targets. Specifically, we first try to allocate as many villagers as possible so that the coverage they provide is fully utilized (Lines 8 to 10). If some villagers remain, it means that allocating them would cause some targets to have more coverage than necessary. We then allocate these villagers to the targets that minimize the wasted coverage (Lines 11 to 12). Finally, we check whether there are enough ranger efforts to cover the remaining needs on all targets (Line 13). Algorithm 1 works in $O(n)$ time to check whether there is a consistent strategy. Formally, we have the following lemma.

Lemma 4.1. *Algorithm 1 returns* True *if and only if there exists a valid defender strategy profile* $(\mathbf{p}, \mathbf{v})$ *such that* $p_{i^*} = p, v_{i^*} = v$ *and* i^* *is the attacker's best response.*

The proof of Lemma 4.1 is deferred to Appendix A.1.

Now that we have the ability to judge whether a consistent strategy exists, for target i^* and fixed defender strategy (p_{i^*}, v_{i^*}) on i^*. For those pairs of (p_{i^*}, v_{i^*}) with a consistent strategy, we would like to find the one that maximizes the coverage $c_{i^*} = \min(p_{i^*} \cdot e^{\mathrm{p}} + v_{i^*} \cdot e^{\mathrm{v}}, 1)$ on i^*. We will use a two-dimensional binary search to find this maximum coverage. To do this efficiently, we need to establish two monotonicity lemmas: Lemmas 4.2 and 4.3.

Lemma 4.2. *Let* i^* *be a target, and let* $(\mathbf{p}, \mathbf{v})$ *be a valid defender strategy profile such that* i^* *is the attacker's best response. Then,* $\forall 0 \leq p \leq p_{i^*}$, $0 \leq v \leq v_{i^*}$ $(v \in \mathbb{N})$, *there is a valid defender strategy profile* $(\mathbf{p}', \mathbf{v}')$ *such that* $p'_{i^*} = p, v'_{i^*} = v$ *and* i^* *is still one of attacker's best responses.*

We present the proof of Lemma 4.2 in Appendix A.2. Intuitively, Lemma 4.2 shows for a target i^* that an attacker will attack with the best response under some defender strategy $(\mathbf{p}, \mathbf{v})$, we can always reduce the resources allocated to i^* while keeping the attacker's best response unchanged.

We then move on to another monotonicity lemma. Rangers and villagers can substitute each other on a specific target provided that they are of the same effectiveness, e.g., one villager can be replaced with $e^{\mathrm{v}}/e^{\mathrm{p}}$ units of ranger effort. However, since a villager can only patrol on one target while a ranger can distribute their efforts among multiple targets, ranger efforts are more flexible, and thus more useful when the two kinds of resources can be converted into the same amount of coverage. Therefore, converting ranger efforts to villagers on target i^* makes it easier to satisfy the coverage needs of the other targets to make i^* the attacker's best response. We formally state this intuition as follows.

Lemma 4.3. *Let* i^* *be a target, and let* $(\mathbf{p}, \mathbf{v})$ *be a valid defender strategy profile such that* i^* *is the attacker's best response. Then, for any* $p \in \mathbb{R}_{\geq 0}, v \in \mathbb{N}$ *such that* $p \cdot e^{\mathrm{p}} + v \cdot e^{\mathrm{v}} = p_{i^*} \cdot e^{\mathrm{p}} + v_{i^*} \cdot e^{\mathrm{v}}$, $p \leq p_{i^*}$ *and* $v \geq v_{i^*}$, *there exists a valid defender strategy profile* $(\mathbf{p}', \mathbf{v}')$ *such that* $p'_{i^*} = p, v'_{i^*} = v$ *and* i^* *is still one of attacker's best responses.*

The proof of Lemma 4.3 is deferred to Appendix A.3.

With the lemmas above, we are ready for our algorithm to compute the maximum defender utility in the RACPP problem approximately. The algorithm is shown in Algorithm 2.

Algorithm 2 enumerates all possible targets i^* as the attacker's best response (Line 2) and tries to find the maximum coverage on i^* as mentioned above. To do this, it first finds the maximum number of villagers v_{i^*} that can be deployed on target i^* such that we can still make target i^* the attacker's best response (Lines 5 to 12). Using the monotonicity established in Lemma 4.2 as well as Algorithm 1, we can use binary search to find such v_{i^*}. Note that as we maximize the villagers used on target i^*, we also maximally substitute any ranger efforts that will be allocated to i^* with villagers. According to Lemma 4.3, this will not make it harder to satisfy the coverage needs of the other targets.

We then fix the number of villagers deployed at i^* and find the maximum possible ranger efforts p_{i^*} on target i^* (Lines 13 to 20). Again by Lemma 4.2, p_{i^*} can also be found using binary search. This pair of (p_{i^*}, v_{i^*}) is guaranteed to be optimal because we first try to maximize v_{i^*}, and then p_{i^*}. Finally, after enumerating each possible $i^* \in T$, the algorithm will return the best defender utility (Lines 21 to 22). Throughout the process, Algorithm 2 makes a total of $O(n \log \frac{M}{\varepsilon})$ calls to Algorithm 1, where M is the maximum absolute value of the input variables. Therefore, Algorithm 2 works in $O(n^2 \log \frac{M}{\varepsilon})$ time.

Algorithm 2: Two-dimensional binary search

Input: Input instance $\mathcal{I}$ and precision ε
Output: Approximate maximum defender utility

1: Let $u_{\mathrm{ans}} \leftarrow -\infty$.
2: **for** $i^* \in \{0, 1, \ldots, n-1\}$ **do**
3:   **if** Algorithm 1 returns False on $(\mathcal{I}, i^*, 0, 0)$ **then**
4:     **continue**
5:   Let $(v_{\mathrm{left}}, v_{\mathrm{right}}) \leftarrow (0, r^{\mathrm{v}})$.
6:   **while** $v_{\mathrm{left}} \leq v_{\mathrm{right}}$ **do**
7:     Let $v_{\mathrm{cur}} \leftarrow \lfloor (v_{\mathrm{left}} + v_{\mathrm{right}})/2 \rfloor$.
8:     **if** Algorithm 1 returns True on $(\mathcal{I}, i^*, 0, v_{\mathrm{cur}})$ **then**
9:       Let $v_{\mathrm{left}} \leftarrow v_{\mathrm{cur}} + 1$.
10:       Let $v_{i^*} \leftarrow v_{\mathrm{cur}}$.
11:     **else**
12:       Let $v_{\mathrm{right}} \leftarrow v_{\mathrm{cur}} - 1$.
13:   Let $(p_{\mathrm{left}}, p_{\mathrm{right}}) \leftarrow (0, r^{\mathrm{p}})$.
14:   **while** $p_{\mathrm{right}} - p_{\mathrm{left}} > \varepsilon$ **do**
15:     Let $p_{\mathrm{cur}} \leftarrow (p_{\mathrm{left}} + p_{\mathrm{right}})/2$.
16:     **if** Algorithm 1 returns True on $(\mathcal{I}, i^*, p_{\mathrm{cur}}, v_{i^*})$ **then**
17:       Let $p_{\mathrm{left}} \leftarrow p_{\mathrm{cur}}$.
18:       Let $p_{i^*} \leftarrow p_{\mathrm{cur}}$.
19:     **else**
20:       Let $p_{\mathrm{right}} \leftarrow p_{\mathrm{cur}}$.
21:   Compute $U^{\mathrm{d}}_{i^*}$ using (p_{i^*}, v_{i^*}).
22:   Let $u_{\mathrm{ans}} \leftarrow \max(u_{\mathrm{ans}}, U^{\mathrm{d}}_{i^*})$.
23: **return** u_{ans}.

Theorem 4.1. *Let the absolute values of the input variables be bounded by* M. *Algorithm 2 generates a valid defender strategy profile* $(\mathbf{p}, \mathbf{v})$ *in* $O(n^2 \cdot \log \frac{M}{\varepsilon})$ *time such that for the optimal defender strategy profile* $(\mathbf{p}^*, \mathbf{v}^*)$,

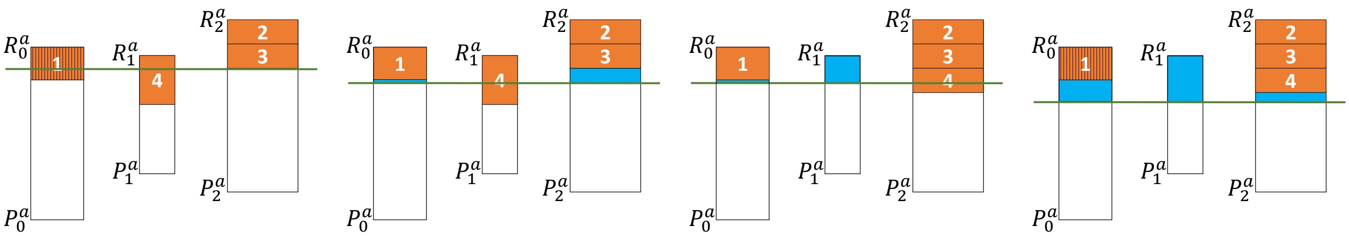
$$u(\mathbf{p}^*, \mathbf{v}^*) - u(\mathbf{p}, \mathbf{v}) < e^{\mathrm{p}} \cdot 2M\varepsilon.$$

The proof of Theorem 4.1 is presented in Appendix A.4.

4.3 A Polynomial Exact Algorithm: Hybrid Waterfilling Algorithm

The Two-Dimensional Binary Search algorithm can solve the RACPP problem to any given accuracy ε. However, it is not able to solve the problem precisely. Therefore, in this section, we study exact polynomial-time algorithms for the RACPP problem. We will present a polynomial-time algorithm, the Hybrid Waterfilling algorithm, that solves the RACPP problem precisely.

The sense of waterfilling has been applied to mathematical models in security domains (Kiekintveld et al. 2009;



(a) Greedy for villagers (b) Waterfilling to a critical point (c) Trigger a **swap** (d) Finish Waterfilling

Figure 1: Example of the process in Algorithm 4. The 3 black rectangles represent 3 targets. The orange parts are the villager coverage and the blue ones are the ranger coverage. The green line is the utility sea level. (a) Target $i^* = 0$ and villager strategy $v_{i^*} = 1$ are given. We greedily allocate the remaining 3 villagers to target $i \in \{1, 2\}$ with the maximum U_i^{a} in the order of target 2, 2, 1. (b) Critical set is $\{1\}$. We distribute ranger efforts by Waterfilling, lowering the sea level, until we reach the critical point when the area of the blue part on target 3 is equal to the area of villager 3 above the sea level. (c) We swap ranger efforts on target 2 and villager 3 on target 2. (d) We proceed with Waterfilling until rangers are used up.

Nguyen et al. 2015). The main challenge caused by the involvement of villagers is deterministic allocation. The efforts of a villager can only be allocated to one target and can not be distributed to multiple targets, which breaks the continuous change of water level in standard waterfilling algorithm (Thomas and Joy 2006).

When there are only rangers, the RACPP problem becomes a standard security game problem, which can be exactly solved by Waterfilling. Whereas when there are only villagers, the RACPP problem can also be solved exactly by greedily allocating each villager to the target with the maximum attacker utility. Our Hybrid Waterfilling algorithm combines these two methods to solve the RACPP problem precisely. We formally state the algorithm as Algorithm 5 in Appendix B. Below, we present the intuition.

Algorithm 5 relies on an important subproblem: given a target i^* that the attacker will attack and a fixed villager strategy v_{i^*} on i^*, how to allocate the remaining villagers and rangers to maximize the defender utility? We use a hybrid method of Greedy and Waterfilling to solve this subproblem. See Fig. 1 for an illustration.

First, we focus only on the villagers, greedily allocating them to targets $i \in T \setminus \{i^*\}$ with the maximum attacker utility U_i^{a} (Fig. 1a). At this time, there is a set of targets that maximize the attacker's utility. We call this set the **critical set**, and the corresponding utility the **utility sea level**. We then distribute ranger efforts in a way similar to Waterfilling, i.e., always allocating them to the critical set to lower the sea level. At some point in this process, it might be possible to exchange a villager and some ranger efforts on two targets, such that the sea level remains unchanged but the villager goes to a target with a larger **width** ($w_i = 1/(R_i^{\text{a}} - P_i^{\text{a}})$) (Fig. 1b), we call this a **critical point**, which triggers a **swap** (Fig. 1c). After the swap, the total width of the critical set decreases, and thus the amount of ranger efforts required to lower a unit of the sea level decreases. We continue this process until all ranger efforts are used up (Fig. 1d). We formally summarize the process as Algorithm 4 in Appendix B. The algorithm works in $O(n^3 \log n)$ time.

With the subproblem solved, we can then solve RACPP. To do this, we first enumerate all $i^* \in T$. For each given i^*, we first use binary search to find the maximum number of villagers that can be allocated to target i^* while ensuring i^* is still the attacker's best response. This is similar to the Two-Dimensional Binary Search algorithm. We then call Algorithm 4 to find the optimal defender utility for the given i^* and v_{i^*}. Finally, we return the maximum defender utility among all i^*. The optimality and time complexity of the algorithm is demonstrated in the following theorem.

Theorem 4.2. *Algorithm 5 generates an optimal valid defender strategy profile* $(\mathbf{p}, \mathbf{v})$ *in* $O(n^4 \log n)$ *time.*

The proof of Theorem 4.2 is deferred to Appendix B. Below, we provide a proof sketch of the theorem.

Proof Sketch of Theorem 4.2. To show the correctness of the algorithm, we first need to show its correctness for the subproblem, i.e., Algorithm 4 generates an optimal valid defender strategy profile $(\mathbf{p}, \mathbf{v})$ for a given target i^* and villager deployment strategy v_{i^*}. Note that in Algorithm 4, the sea level u_{cur} lowers continuously as we distribute ranger efforts. For a fixed sea level u_{cur} in the Waterfilling process, Algorithm 4 generates a strategy profile that ensures that i^* is the attacker's best responses with utility u_{cur}. In this strategy profile, we define the **wasted villager coverage** $\text{scw}(u_{\text{cur}})$ as the coverage provided by villagers that are not needed to ensure that i^* is the attacker's best response.

We will show that the wasted villager coverage is minimized at each sea level u_{cur}, which effectively means that the amount of ranger efforts required to reach this sea level is minimized. This is done in two steps: **(i).** We show that given sea level u_{cur}, we cannot redistribute the ranger efforts and at most one villager to reduce the wasted villager coverage. **(ii).** We show that the wasted villager coverage is minimized at each sea level u_{cur}. For step (i), the calculation of the critical points in Algorithm 4 ensures it. For step (ii), we show by an adjustment argument that if (i) holds, then the structure of the wasted villager coverage ensures that redistributing more villagers cannot reduce the wasted villager coverage, either. This completes the proof of the correctness of Algorithm 4. The correctness of Algorithm 5 then follows from the correctness of Algorithm 4 and the correctness of the binary search part, which is similar to Algorithm 2.

For the complexity, since Algorithm 1 works in $O(n)$, the binary search part of Algorithm 5 works in $O(n^2 \log M)$ time, where M is the maximum absolute value of the input variables. For the procedure of the subproblem in Algorithm 4, note that each swap operation causes a villager to be moved to a target with a larger width, and only $O(n)$ villagers can possibly be moved. Therefore, the number of swaps, i.e., the number of iterations of Algorithm 4 is $O(n^2)$. Using a priority queue to simulate the procedure, each iteration takes $O(n \log n)$ time. Therefore, the total time complexity of Algorithm 4 is $O(n^3 \log n)$. Assuming M is polynomially bounded by n, the total time complexity of Algorithm 5 is $O(n^4 \log n)$. This concludes the proof sketch. □

5 Extensions for Practical Constraints

In practice, the actual defense effectiveness varies with real-world factors. For instance, geographical features like vegetation and slope can affect the defense effectiveness, which results in differences in defense effectiveness on different targets. Moreover, individual factors, like the domain knowledge and experience of the person, can also influence defense effectiveness. Due to their lack of training, these practical factors have a greater impact on the villagers. Therefore, in this section, we consider two generalized versions of the RACPP problem, where the villagers' defense effectiveness varies with the targets and the villagers, respectively. Interestingly, for the former, our algorithms in Section 4 can be adapted to solve the problem exactly in polynomial time. However, in the latter case, the problem becomes NP-hard.

RACPP with Target-Specific Effectiveness. We first consider the case where the defense effectiveness of villagers varies with the targets. Specifically, we redefine villagers' defense effectiveness as $\mathbf{e}^{\mathrm{v}}$, where e_i^{v} represents the effectiveness of one villager on target i.

Both algorithms in Section 4 can be naturally adapted to solve RACPP with target-specific effectiveness. For simplicity, we only present the adapted version of Algorithm 2 here.

Recall that Algorithm 2 is based on a two-dimensional binary search that needs to check whether a consistent strategy exists for a given target i^* and fixed defender strategy (p_{i^*}, v_{i^*}) on i^* (Algorithm 1). In this new setting, we modify this procedure to Algorithm 6 stated in Appendix C.

Like Algorithm 1, the general idea of Algorithm 6 is to greedily distribute the resources to other targets to ensure that the attacker's utility is no more than $U_{i^*}^{\mathrm{a}}$. The main difference is that the coverage generated by villagers on each target is now different. For each villager, the algorithm greedily allocates them to the target, which maximizes the coverage each villager can cover. Finally, the algorithm checks whether there are enough ranger efforts as in Algorithm 1. The whole process can be implemented in $O(n)$ time. Formally, we have Lemma 5.1.

Lemma 5.1. *Algorithm 6 returns* True *if and only if there exists a valid defender strategy profile* $(\mathbf{p}, \mathbf{v})$ *such that* $p_{i^*} = p, v_{i^*} = v$ *and* i^* *is the attacker's best response.*

The proof of Lemma 5.1 is deferred to Appendix D.1. Algorithm 2 in TDBS Algorithm can stay unchanged in the new setting, and the time complexity is still $O(n^2 \log \frac{M}{\varepsilon})$.

RACPP with Villager-Specific Effectiveness. We then consider the case where the defense effectiveness of villagers varies with the villagers. Let $V = \{0, 1, \cdots, r^{\mathrm{v}} - 1\}$ be the set of villagers. We redefine villagers' defense effectiveness as a vector $\mathbf{e}^{\mathrm{v}}$, where e_j^{v} represents the effectiveness of villager $j \in V$ on all targets.

Interestingly, the problem becomes NP-hard in this setting. We prove that RACPP with villager-specific effectiveness is NP-hard by reducing the partition problem (Hayes 2002) to it. The details are deferred to Appendix D.2.

Theorem 5.1. *Computing the maximum defender utility and the optimal valid strategy profile of the RACPP problem with villager-specific effectiveness is NP-hard.*

6 Experiments

In this section, we conduct numerical experiments on synthetic data to evaluate the performance of our algorithms in practice. We implement our Two-Dimensional Binary Search algorithm (TDBS) and Hybrid Waterfilling algorithm (HW), along with a benchmark algorithm using Gurobi's (Gurobi Optimization, LLC 2023) Mixed Integer Linear Programming (MILP) solver. Since all of the implemented algorithms have guaranteed solution quality, we observe similar performance in terms of the defender's utility from all of them. Therefore, we focus on the runtime of these algorithms. The experiments are conducted on a server with an Intel Xeon E5-2683 v4 CPU and 269.5GB RAM.

Experiment setup. We evaluate the algorithms with different combinations of $(n, r^{\mathrm{p}}, r^{\mathrm{v}})$. For each combination, we randomly generate e^{p}, e^{v}, $\mathbf{R}^{\mathrm{d}}$, $\mathbf{P}^{\mathrm{d}}$, $\mathbf{R}^{\mathrm{a}}$, $\mathbf{P}^{\mathrm{a}}$ with $0 < ev < ep < 1$, $R_i^{\mathrm{d}}, R_i^{\mathrm{a}} \in [0, 10)$ and $P_i^{\mathrm{d}}, P_i^{\mathrm{a}} \in [-10, 0)$ and record the runtime of the algorithms. We report the mean and the standard deviation of the runtime over 30 runs for each combination. The precision of TDBS is set to 10^{-3}. If a single run of an algorithm exceeds 7200 seconds, we interrupt it and record the runtime as 7200 seconds.

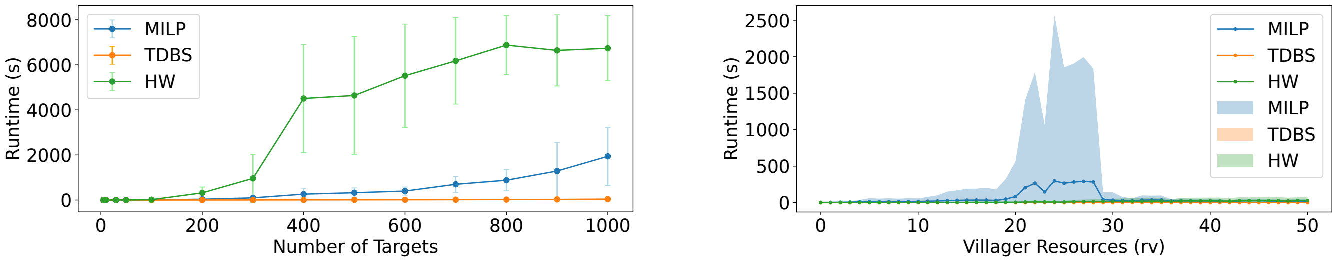
Performance with different n. We let $n = \{5, 10, 30, 50, 100, 200, 300, 400, 500, 600, 700, 800, 900, 1000\}$, and let r^{p} and r^{v} be $\lfloor \frac{n}{2} \rfloor$. As shown in Fig. 2a, the runtime of TDBS is significantly lower than the other two algorithms. Although it is not an exact algorithm, the results with precision 10^{-3} are accurate enough for practical applications with the guarantee proven in Theorem 4.1.

Performance with different $r^{\mathrm{p}}, r^{\mathrm{v}}$. We then fix $n = 100$ and let $r^{\mathrm{p}} = r^{\mathrm{v}}$ be $\{0, 1, \ldots, 50\}$. As shown in Fig. 2b, both of our proposed algorithms' runtimes are stable with different r^{p} and r^{v}. However, the performance of MILP is highly unstable due to its lack of time complexity guarantee.

Additional experiments are presented in Appendix E.

7 Case Study on Anti-poaching

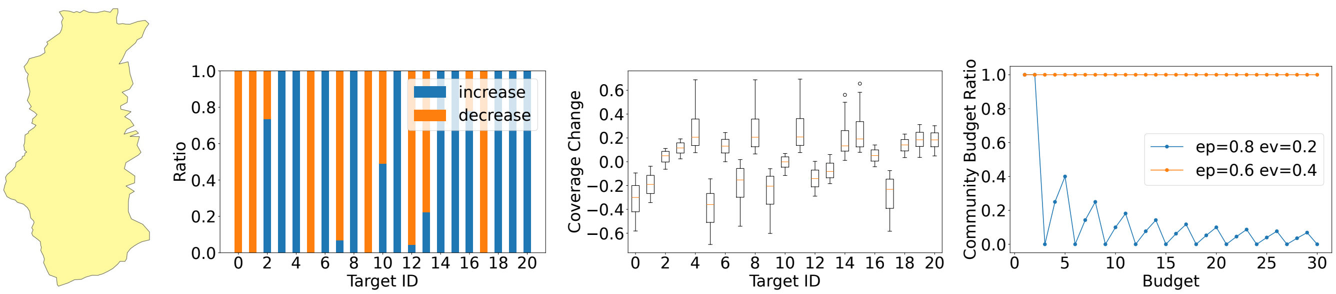
We have applied RACPP to a protected area in Northeast China, home to the Manchurian tiger. To protect the raw



(a) runtime with different n and $r^{\mathrm{p}} = r^{\mathrm{v}} = \lfloor \frac{n}{2} \rfloor$

(b) runtime with $n = 100$ and different $r^{\mathrm{p}}, r^{\mathrm{v}}$

Figure 2: Average runtime of MILP, TDBS, and HW over 30 runs under different combinations of $(n, r^{\mathrm{p}}, r^{\mathrm{v}})$. The error bars indicate the standard deviation. The shaded areas represent the ranges from the minimum to the 97th percentile. We limit the maximum runtime for MILP in Fig. 2b to 7200 seconds.



(a) Contour (b) Ratio of increase and decrease (c) Coverage change (d) Budget allocation

Figure 3: The contour of the studied forest farm and case study results regarding advice on strategies and budget allocation.

data which includes past patrol allocation and animal density, we only report the information that can be made public based on discussions with local agencies. The contour of the protected area is shown in Fig. 3a. The protected area is divided into 21 2km×2km regions in the last three patrol seasons for patrol planning, and we naturally use these regions as targets, i.e., $T = \{0, 1, \ldots, 20\}$. Ranger resources r^{p} and villager resources r^{v} are provided by local agencies. Besides, since roe deer and sika deer are the main prey of Manchurian tigers in that area, we use a weighted distribution of multiple species including roe deer, sika deer, and wild boars to estimate the reward for the poacher R_i^{a}, which is shown in Table 1 in Appendix F. We set $P_i^{\mathrm{d}} = -R_i^{\mathrm{a}}$. Since defenders care less about where they find snares or catch attackers and attackers face the same amount of fine wherever they are caught, $-P_i^{\mathrm{a}}$ and R_i^{d} are set to a fixed number 10.

In this experiment, we calculate and compare the optimal patrol strategy with the current one. First, we obtain the villager allocation strategy from local agencies and process last season's ranger patrol records to determine their strategy. Specifically, we calculate the total length of patrol routes within each target to derive the distribution of ranger efforts and, combined with ranger resources r^{p}, estimate the resources allocated to each target. Since the exact values of e^{p} and e^{v} are unknown, we enumerate their values in $0.1, 0.2, \ldots, 0.9$ with $e^{\mathrm{p}} \geq e^{\mathrm{v}}$, resulting in 45 settings. After generating optimal defender strategies and total coverage for each target, we compare them to the current strategy in the protected area and summarize the findings in Fig. 3b. The length of the orange bar shows how many $(e^{\mathrm{p}}, e^{\mathrm{v}})$ settings suggest reducing coverage on that target. For instance, for target 0, the optimal coverage is lower than the current level across all settings, while target 18 consistently requires increased coverage. Fig. 3c shows the distribution of coverage changes for all targets. Defender utility is expected to improve 25. 9% - 152. 6%, with an average of 83. 1%.

When the total budget for recruiting rangers and villagers increases, should we use the extra money to recruit more rangers or more villagers? The current cost ratio of one ranger versus one villager is 3:1. We calculate the optimal plan of allocating the extra money with two different $(e^{\mathrm{p}}, e^{\mathrm{v}})$ settings when the budget increase ranges from 1 unit to 30 units and show the results in Fig. 3d. When $e^{\mathrm{p}} = 0.8$ and $e^{\mathrm{v}} = 0.2$, prioritizing recruitment of rangers is more cost-efficient. In contrast, when $e^{\mathrm{p}} = 0.6$ and $e^{\mathrm{v}} = 0.4$, we should spend all budgets on villagers. Additional results for other settings are shown in Appendix F.

We run additional experiments that take into account the terrain information as it is relatively easier to find snares on targets with higher slope variance according to domain experts. The results are shown in Appendix F.

We plan to collaborate with the local forest bureau in Northeast China to adjust future patrol resource allocation based on the case study findings. The planned work includes refining the range of e^{p} and e^{v} through a poaching detection competition where the locations of the poaching tools are known and thus offering more precise recommendations.

References

Basilico, N.; Celli, A.; De Nittis, G.; and Gatti, N. 2017. Coordinating multiple defensive resources in patrolling games with alarm systems. In *Autonomous Agents and MultiAgent Systems*, 678–686.

Bondi, E.; Oh, H.; Xu, H.; Fang, F.; Dilkina, B.; and Tambe, M. 2020. To signal or not to signal: Exploiting uncertain real-time information in signaling games for security and sustainability. In *AAAI*, volume 34, 1369–1377.

Bradley, S. P.; Hax, A. C.; and Magnanti, T. L. 1977. *Applied mathematical programming*. Addison-Wesley.

Castiglioni, M.; Marchesi, A.; and Gatti, N. 2021. Committing to correlated strategies with multiple leaders. *Artificial Intelligence*, 300: 103549.

Conitzer, V.; and Sandholm, T. 2006. Computing the optimal strategy to commit to. In *ACM conference on Electronic commerce*, 82–90.

Danoff-Burg, J. A.; and Ocañas, R., Alejandrina. 2022. Individual and community-level impacts of the unarmed all-women Black Mamba Anti-Poaching Unit. *Zoo Biology*, 41(5): 479–490.

Fang, F.; Stone, P.; and Tambe, M. 2015. When Security Games Go Green: Designing Defender Strategies to Prevent Poaching and Illegal Fishing. In *IJCAI*, volume 15, 2589–2595.

Gan, J.; Guo, Q.; Tran-Thanh, L.; An, B.; and Wooldridge, M. 2019. Manipulating a learning defender and ways to counteract. *Advances in Neural Information Processing Systems*, 32.

Gill, C.; Weisburd, D.; Telep, C. W.; Vitter, Z.; and Bennett, T. 2014. Community-oriented policing to reduce crime, disorder and fear and increase satisfaction and legitimacy among citizens: A systematic review. *Journal of experimental criminology*, 10: 399–428.

Gurobi Optimization, LLC. 2023. Gurobi Optimizer Reference Manual.

Gutierrez, J.; Kowara, S.; Kraus, S.; Steeples, T.; and Wooldridge, M. 2023. Cooperative concurrent games. *Artificial Intelligence*, 314: 103806.

Hayes, B. 2002. Computing science: The easiest hard problem. *American Scientist*, 90(2): 113–117.

Huang, T.; Shen, W.; Zeng, D.; Gu, T.; Singh, R.; and Fang, F. 2020. Green security game with community engagement. *arXiv preprint arXiv:2002.09126*.

Johnson, M.; Fang, F.; and Tambe, M. 2012. Patrol strategies to maximize pristine forest area. In *AAAI*, volume 26, 295–301.

Karp, R. M. 2010. *Reducibility among combinatorial problems*. Springer.

Kiekintveld, C.; Jain, M.; Tsai, J.; Pita, J.; Ordónez, F.; and Tambe, M. 2009. Computing optimal randomized resource allocations for massive security games. *AAMAS-09*.

Linkie, M.; Martyr, D. J.; Harihar, A.; Risdianto, D.; Nugraha, R. T.; Maryati; Leader-Williams, N.; and Wong, W.-M. 2015. Safeguarding Sumatran tigers: evaluating effectiveness of law enforcement patrols and local informant networks. *Journal of Applied Ecology*, 52(4): 851–860.

Massé, F.; Gardiner, A.; Lubilo, R.; and Themba, M. N. 2017. Inclusive anti-poaching? Exploring the potential and challenges of community-based anti-poaching. *South African Crime Quarterly*, 60: 19–27.

Mc Carthy, S. M.; Tambe, M.; Kiekintveld, C.; Gore, M.; and Killion, A. 2016. Preventing illegal logging: Simultaneous optimization of resource teams and tactics for security. In *AAAI*, volume 30.

McCARTHY, S. M.; Laan, C. M.; Wang, K.; Vayanos, P.; Sinha, A.; and Tambe, M. 2018. The price of usability: Designing operationalizable strategies for security games. In *IJCAI*. IJCAI.

Mutzari, D.; Gan, J.; and Kraus, S. 2021. Coalition formation in multi-defender security games. In *AAAI*, volume 35, 5603–5610.

Nguyen, T.; and Xu, H. 2019. Imitative Attacker Deception in Stackelberg Security Games. In *IJCAI*, 528–534.

Nguyen, T. H.; Fave, F. M. D.; Kar, D.; Lakshminarayanan, A. S.; Yadav, A.; Tambe, M.; Agmon, N.; Plumptre, A. J.; Driciru, M.; Wanyama, F.; et al. 2015. Making the most of our regrets: Regret-based solutions to handle payoff uncertainty and elicitation in green security games. In *GameSec*, 170–191. Springer.

Nubani, L.; Fierke-Gmazel, H.; Madill, H.; and De Biasi, A. 2023. Community Engagement in Crime Reduction Strategies: A Tale of Three Cities. *Journal of Participatory Research Methods*, 4(1).

Shen, W.; Chen, W.; Huang, T.; Singh, R.; and Fang, F. 2020. When to follow the tip: Security games with strategic informants. In *IJCAI*.

Shen, W.; Han, M.; Chen, W.; Huang, T.; Singh, R.; Xu, H.; and Fang, F. 2024. An Extensive Study of Security Games with Strategic Informants. *Artificial Intelligence*, 104162.

Sjöstedt, M.; Sundström, A.; Jagers, S. C.; and Ntuli, H. 2022. Governance through community policing: What makes citizens report poaching of wildlife to state officials? *World Development*, 160: 106065.

Stackelberg, H. v. 1934. Marktform und gleichgewicht. *(No Title)*.

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

Thomas, M.; and Joy, A. T. 2006. *Elements of information theory*. Wiley-Interscience.

Wilson-Holt, O.; and Roe, D. 2021. Community-Based Approaches to Tackling Illegal Wildlife Trade—What Works and How Is It Measured? *Frontiers in Conservation Science*, 2: 765725.

Xu, H.; Wang, K.; Vayanos, P.; and Tambe, M. 2018. Strategic coordination of human patrollers and mobile sensors with signaling for security games. In *AAAI*, volume 32.