# Preventing Illegal Logging: Simultaneous Optimization of Resource Teams and Tactics for Security

Sara Marie Mc Carthy  
University of Southern California  
saramarm@usc.edu

Milind Tambe  
University of Southern California  
tambe@usc.edu

Christopher Kiekintveld  
University of Texas at El Paso  
cdkiekintveld@utep.edu

Meredith L. Gore  
Michigan State University  
gorem@msu.edu

Alex Killion  
Michigan State University  
killiona@msu.edu

## ABSTRACT

Green security – protection of forests, fish and wildlife – is a critical problem in environmental sustainability. We focus on the problem of optimizing the defense of forests against illegal logging, where often we are faced with the challenge of teaming up many different groups, from national police to forest guards to NGOs, each with differing capabilities and costs. This paper introduces a new, yet fundamental problem: Simultaneous Optimization of Resource Teams and Tactics (SORT). SORT contrasts with most previous game-theoretic research for green security – in particular based on security games – that has solely focused on optimizing patrolling tactics, without consideration of team formation or coordination. We develop new models and scalable algorithms to apply SORT towards illegal logging in large forest areas. We evaluate our methods on a variety of synthetic examples, as well as a real-world case study using data from our on-going collaboration in Madagascar.

1This work has been previously published in AAAI-16

## General Terms

Algorithms, Security

## Keywords

Game theory, Multi-Agent Systems

## 1. INTRODUCTION

Illegal logging is a major problem for many developing countries. The economic and environmental impacts are severe, costing up to USD $30 billion annually and threatening ancient forests and critical habitats for wildlife [18]. As a result, improving the protection of forests is of great concern for many countries [2, 17].

Unfortunately in developing countries, budgets for protecting forests are often very limited, making it crucial to allocate these resources efficiently. We focus on deploying resources to interdict the traversal of illegal loggers on the network of roads and rivers around the forest area. However, we must first choose the right security team for interdiction; there are many different organizations that may be involved — from local volunteers, to police, to NGO personnel — each differing in their interdiction effectiveness (individual or jointly with others), and with varying costs of deployment. This results in a very large number of resource teams and allocation strategies per team with varying effectiveness that could be deployed within a given budget. Our challenge is to simultaneously select the best team of security resources and the best allocation of these resources.

Over the past decade, game-theoretic approaches, in particular security games, have become a major computational paradigm for security resource allocation [16, 10]. The sub-area of security games relevant to this paper is at the intersection of green security (i.e., protection of forests, fish and wildlife) [4, 9, 13], and network security games (NSG) (i.e., interdiction of adversaries on transportation networks) [7]. However, previous work in these areas suffer from two limitations relevant to our problem. First, they only consider the tactical allocation of a given team of resources, without considering the more strategic question of how to choose the right team. Indeed, the fact that the tactical question is already computationally challenging emphasizes the difficulty of our problem, which requires evaluating the effectiveness of many teams to select the right one. Second, previous work has mostly failed to consider heterogeneous teams, with varying individual and joint effectiveness.

To address these challenges we make the following key contributions. First, we introduce Simultaneous Optimization of Resource Teams and Tactics (SORT) as a new fundamental research problem in security games that combines strategic and tactical decision making and provide a formal model of a SORT problem in an NSG. Second, at the strategic level, we introduce FORTIFY (Forming Optimal Response Teams for Forest safety), a scalable algorithm for solving this problem. FORTIFY uses a hierarchical approach to abstract NSGs at varying levels of detail, providing bounds
on the value of different teams of resources to speed up the search for the optimal team. Third, at the tactical level, we generalize previous methods for optimizing tactical resource allocation in network security games to account for heterogeneous teams with varying capabilities. Lastly, we present experimental results on synthetic problems, as well as problem instances obtained from joint work with NGOs engaged in forest protection in Madagascar.

2. MOTIVATING DOMAIN AND GAME MODEL

Forests in Madagascar are under great threat, with valuable hardwood trees such as rosewood and ebony being illegally harvested at an alarming rate. There is broad interest in improving forest protection via patrolling from different groups, e.g., NGOs, the Madagascar National Parks, local police and community volunteers. However, there is very limited coordination among these groups. As a result there is limited attention paid at the strategic level to optimize the selection of the right team of resources from among these groups, and the tactical level to optimally allocate the resulting team’s resources. Our model is designed to address these problems.

![Illegal logging in progress in at risk area of Madagascar, images provided by our partnering NGOs working in the area.](image)

**Figure 1:** Illegal logging in progress in at risk area of Madagascar, images provided by our partnering NGOs working in the area.

**Model:** We now describe our model for SORT. At the tactical level, the decision of how to optimally allocate a team is a NSG problem. We model the physical space using a graph $G = (N,E)$, consisting of source nodes $s \in S \subset N$, target nodes $t \in T \subset N$, and intermediate nodes. The attacker (illegal loggers) acts by traversing a path from a source $s_i$ to a target node $t_i$. For illegal logging, the $s_i$ may be attacker’s originating villages and $t_i$ may be areas containing valuable trees. Each target node $t_i$ has a payoff value that is domain dependent. Based on the research of collaborating domain experts in Madagascar, these depend on the density of valuable trees in a particular area, and the distance to the coast (for export) and local villages.

Previous models in NSGs assume a defender limited to homogenous resources with no probability of failure, no joint effectiveness and the ability to only cover a single edge [6, 15]. This is insufficient to capture the complexities present in the illegal logging domain, and so we present a new model of the defender for green NSGs.

The defender conducts patrols in the network to interdict the attacker by placing resources on edges of the graph, as shown in Figure 2. The defender has $K$ types of resources each of which can conduct a patrol along $L_k$ connected edges. Multiple environmental factors can cause a resource to fail in detecting an attacker, such as limited visibility or collusion with the adversaries. We model this using an interdiction probability $P_b$ for each resource type. The defender has a total budget $B$; each resource type has a cost $b_k$, and a team consists of $m_k$ resources of type $k$ for $k = 1 \ldots K$. Multiple resources placed on a single edge results in a higher probability of detecting the attacker, which models coordination among resources.

A defender pure strategy $X_i$ is an allocation of all resources in a given team to a set of edges of the graph, satisfying the connectedness and length constraints for each resource. An attacker pure strategy $A_j$ is any path starting at a source node $s_j$ and ending at a target node $t_j$. Figure 2 shows three attacker pure strategies. Although these strategies take the shortest path from source to target, it is not a requirement in the game. The allocation of one resource of size $L_e = 2$ is shown, which intersects paths $i$ and $j$. The attacker and defender can play mixed strategies $a$ and $x$, i.e., probability distributions over pure strategies. The probability of detecting an attacker on edge $e$ if the defender follows a pure strategy $X_i$, allocating $m_{k,e}$ number of resources of type $k$ to edge $e$ is given in Equation 1.

$$P(e, X_i) = 1 - \prod_{1}^{K} (1 - P_b)^{m_{k,e}}$$

The total probability that a defender pure strategy $X_i$ protects against an attacker pure strategy $A_j$ is given by the probability intersection function in Equation 2, where we take the product over all the edges in the attack path.

$$P(X_i, A_j) = 1 - \prod_{e \in A_j} (1 - P(e, X_i))$$

The attacker obtains a payoff of $\tau(t_i)$ if he is successful in reaching a target, and a payoff of zero he is caught. We assume a zero sum model, so the defender receives a penalty opposite of the attacker’s payoff. For this zero-sum game, the optimal defender mixed strategy is the well-known

| $G(N,E)$ | Graph representing security domain |
| $G_c$ | Compact Graph representing security domain |
| $\tau(t_i)$ | Payoff of the $i^{th}$ target $t_i$ |
| $K$ | Number of defender resource types |
| $L_k$ | Number of edges covered by the $k^{th}$ resource type |
| $b_k$ | Cost of the $k^{th}$ resource type |
| $P_b$ | Detection probability of resource type $k$ |
| $B$ | Total budget for the team |
| $m_k$ | Number of defender resources of type $k$ |
| $X = \{X_i\}$ | Set of defender pure strategies |
| $x = \{x_i\}$ | Defender’s mixed strategy over $X$ |
| $A = \{A_j\}$ | Set of attacker pure strategies |
| $a = \{a_j\}$ | Attacker’s mixed strategy over $A$ |
| $U_d(X_i, a)$ | Defender utility playing $X_i$ against $a$ |
| $U_a(x, A_j)$ | Attacker utility playing $A_j$ against $x$ |

**Table 1:** Notation and Game Description

![Figure 2: Pure strategies for the Defender (bold line) and Attacker (dashed line going from $s$ to $t$).](image)
minimax strategy. The game value is denoted \( F(\lambda) \), and is function of a team of resources \( \lambda \) selected from some set of resources \( R \). The strategic aspect of the SORT problem can be formulated as the optimization problem in Equation 3, where the utility \( F(\lambda) \) is maximized subject to budgetary constraints.

\[
\max_{\lambda \in R} \left\{ F(\lambda) : \sum_{k \in \lambda} b_k \leq B \right\} \tag{3}
\]

**Theorem 1** The SORT problem is NP-Hard even if we can evaluate \( F(\lambda) \) in constant time. The proof uses a reduction to knapsack. All absent proofs are located in the appendix.\(^2\)

### 3. FORTIFY: THE HIERARCHICAL SEARCH

In NSGs \( F(\lambda) \) is computationally difficult to calculate, because it requires finding the optimal tactical allocation to assess the utility of a given team \( \lambda \). Since there are exponentially many possible teams, the sequential approach of evaluating \( F(\lambda) \) exactly for every team and picking the best one is impractical. Instead, in our approach to SORT, we integrate the analysis of the strategic and tactical aspects of the problem to search the space of teams much more efficiently. We use fast methods to quickly evaluate upper bounds on the utilities for specific teams. Using these bounds we select the most promising team to evaluate in more detail, iteratively tightening the bounds as the search progresses until the optimal team is identified.

FORTIFY uses a three layer hierarchical representation NSG to evaluate the performance of teams at different levels of detail. Starting from the full representation of the game, each layer abstracts away additional details to approximate the game value \( F(\lambda) \). We call the bottom later without any abstraction the Optimal Layer, the Reduced Layer is in the middle, and the Compact Layer is the most abstract.

FORTIFY is described in Algorithm 1 and Figure 3. Initially (line 1), we enumerate all teams \( \Lambda \), that maximally saturate the cost budget \( B \) (so that no additional resources can be added to the team). Each team \( \lambda \in \Lambda \) proceeds through the layers of the hierarchy by being promoted to the next layer; the first team to make it through all layers is the optimal team. When a team is promoted to a new layer, it is evaluated to compute a tighter upper bound on the value based on the abstraction specified for that layer.

---

\(^2\)Supplemental material and proofs are available at http://teamcore.usc.edu/papers/2016/SortAppendix.pdf

---

At the start of the algorithm we have no information on the team values, so each team is evaluated and ordered based on the Compact Layer (line 2-3). Next, the team with the highest bound is promoted to the Reduced Layer (line 9). This team can then be again promoted to the Optimal Layer or another team can be promoted to the Reduced Layer. The next team to be promoted is always the one with the highest current upper bound on the value, regardless of which layer it is in. When a team is evaluated in the Optimal Layer we know the true value of the team, so if this value is higher than the upper bounds on all remaining teams this team must be optimal (line 8), and the algorithm terminates.

---

**Defender Oracle:** The defender oracle returns the best re-
spence strategy $X_1$ to add to the MiniMax LP. The objective is to maximize the utility expressed in Equation (5), given an input distribution $a$ over the current attacker strategies $A$, where $a_j$ the probability of attacker taking path $A_j$.

$$U_d(X_i, a) = - \sum_j a_j (1 - P(X_i, A_j)) \tau(t_j)$$ (5)

A pure strategy implies a single allocation of the given team’s resources. Resources are allocated by setting the binary decision variables $\lambda_{m,e}^k \in \{0, 1\}$ which corresponds to the $m$th resource of type $k$ being allocated to edge $e$. Our contributions formalize the constraints needed to accommodate arbitrary path coverage as well as failure probability. Path constraints are enforced with $\sum_e \lambda_{m,e}^k = L_k$ and in Equations (6-7). Equation (6) ensures every allocated edge is connected to at least one other edge. Since the number of nodes in any path of length $L_k$ should be $L_k + 1$, Equation (7) counts the number of nodes which are either a source or target of allocated edges, making sure not to double count nodes which belong to multiple edges. $\lambda_{m,m}^k \in \{0, 1\}$ and equals 1 if a node $n$ is the source or target of any allocated edge $\lambda_{e,m}^k = 1$.

$$\lambda_{m,e}^k \leq \sum_{e_{1,\in (n)}} \lambda_{m,e}^k + \sum_{e_{\in (n)}} \lambda_{m,e}^k \quad \text{s.t.} \quad n \leftarrow \text{source}(e) \cup \text{target}(e) \quad \sum_n \lambda_{m,n}^k = L_k + 1$$ (6)

$$\lambda_{m,n}^k \leq \lambda_{m,m}^k \quad \text{s.t.} \quad n \leftarrow \text{source}(e) \cup \text{target}(e) \quad \sum_n \lambda_{m,n}^k = L_k + 1$$ (7)

**Attacker Oracle:** The attacker oracle computes a best response strategy $A_j$ which maximizes his utility (Equation 8), playing against the defender mixed strategy $X$. An optimization problem in the form of (8-9) is solved for each target $t_j$; the best path for each target is computed and the target with the highest utility is chosen. The decision variables $\gamma_e \in \{0, 1\}$ are binary and correspond to edges $e \in A_j$.

$$U_a(x, A_j) = \sum_i x_i (1 - P(X_i, A_j)) \tau(t_j)$$ (8)

$$\sum_{\in (n)} \gamma_e = 1; \quad \sum_{\in (n)} \gamma_e = 1; \quad \sum_{\in (n)} \gamma_e = \sum_{\in (n)} \gamma_e \quad n \leftarrow \text{source} \quad n \leftarrow \text{target}$$ (9)

Exactly solving the Attacker Oracle is computationally expensive. Therefore, in line 6, we introduce a new **Linear Attacker Oracle** approximation to quickly generate an approximate best response. Here the probability intersection function is approximated with an additive linear function, given in equation 10, so we can write the oracle as an LP.

**Algorithm 2: Optimal Layer($\lambda$)**

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Initialize $X, A$</td>
</tr>
<tr>
<td>2</td>
<td>do:</td>
</tr>
<tr>
<td>3</td>
<td>$(U^*_d X, a) \leftarrow \text{MiniMax}(X, A)$</td>
</tr>
<tr>
<td>4</td>
<td>$X \leftarrow \text{DefenderOracle}(a)$</td>
</tr>
<tr>
<td>5</td>
<td>$X \leftarrow X \cup {X}$</td>
</tr>
<tr>
<td>6</td>
<td>$A_j \leftarrow \text{LinearAttackerOracle}(X)$</td>
</tr>
<tr>
<td>7</td>
<td>if $U_a(x, A_j) \cdot U_a(x, A_k) \leq \epsilon$ then</td>
</tr>
<tr>
<td>8</td>
<td>$A_j \leftarrow \text{AttackerOracle}(x)$</td>
</tr>
<tr>
<td>9</td>
<td>$A \leftarrow A \cup {A_j}$</td>
</tr>
<tr>
<td>10</td>
<td>until convergence then return $(U^*_d X)$</td>
</tr>
</tbody>
</table>

(In the attacker oracle, the value of $P(e, X_i)$ does not need to be approximated, as it does not depend on attacker’s decision variables, but rather on defender’s variables and thus is calculated outside the attacker oracle.)

$$P(X_i, A_j) = \sum_{e \in A_j} P(e, X_i)$$ (10)

In the event that the approximation steps fail to generate a strategy that increases the oracle’s expected utility (line 7), the oracle computes the optimal solution as a final step (line 8) to ensure that the algorithm converges to the true game value.

5. **COMPACT LAYER**

The compact layer uses an abstract representation of the game model which reduces the problem in two ways: (1) the attacker is restricted to using only a subset of possible paths, (2) the defender chooses to allocate resources directly to attacker paths rather than edges in the graph.

Formally, the compact layer constructs a new graph $G^c(N^c, E^c)$ in Algorithm 3, where the attacker paths are represented by nodes $N^c$ of the graph. We describe this using an example, transforming part of the graph from Figure 2 into it’s compact representation in Figure 4. To choose the subset of attacker paths, for each source-target pair of nodes we (1) calculate the min-cut for each target, and (2) find the shortest possible paths from source to target that go through each of the min-cut edges (lines 1-6). The three attacker paths $A_1, A_2$ and $A_3$ in Figure 2 are among several calculated from the min-cut of the graph. These three paths become nodes $i, j, k$ and $L_k$, respectively, in the new compact graph. In order for a resource to cover paths $i$ and $j$ it’s path coverage $L_k$ must be at least as large as the minimum separation distance between the paths, plus any edges required to intersect the paths. We define this as $D(i, j)$, which is 3 for the example in Figure 2. Edges are added between any nodes $i$ and $j$ with weight $D(i, j)$, equal to the $L_k$, required for each corresponding path. These distances are calculated using Dijkstra’s algorithm, and no edges are added between two nodes if the distance is greater than the largest path coverage of any defender resource.

The defender can choose to cover any subset of nodes in $G^c$ with a resource of type $k$ as long as the induced subgraph has the property that (1) the subgraph is fully connected and (2) all edges have weight less than $L_k$. For example, the three paths in Figure 4 (i-j-k) can be all covered by a
The compact layer solves this abstract representation of the game for a single team. The problem is decomposed into master MiniMax and a single Defender Oracle. There is no oracle for the attacker, as the attacker’s strategy space is enumerated using the compact graph and fixed at the start of the algorithm. The game value is calculated using Algorithm 4. The compact graph and subset of attacker paths are first generated (line 1). The attacker’s mixed strategy is initialized with a uniform distribution (line 2). The compact defender oracle continuously adds strategies to the master LP until the defender’s value cannot be improved. Convergence occurs when the oracle can no longer find a strategy to add that will improve the defender’s utility (line 7).

**Compact Defender Oracle** The same objective function (Equation 5) is maximized, however the constraints are modified to reflect the compact game representation. \( P(X_i, A_j) \) is linearly approximated by Equation 11 and is capped at 1. Here, we want to conservatively over-estimate the defender’s interdiction probability to ensure that the compact layer returns an upper bound. Therefore, when a defender resource covers a node in \( G_c \), we assume that the corresponding attacker path is interdicted by the entire entire patrol of length \( L_k \) of that resource. The probability of catching the attacker on the compact graph is set to \( (1 - (1 - P_j)^L_k) \).

The defender chooses to allocate the \( m \)th resource of type \( k \) to a node \( j \), corresponding to attacker path \( A_j \) by setting the decision variables \( \eta_{j,m}^k \in \{0,1\} \).

\[
P(X_i, A_j) = \sum P_k \eta_{j,m}^k \quad D(i,j)\eta_{i,m}^k\eta_{j,m}^k \leq L_k \quad (11)
\]

**Lemma 1** Playing against the same attacker strategy, the **Optimal Defender Oracle’s strategy space** is a subset of the **Compact Defender Oracle strategy space**.

**Lemma 2** If the **Compact Oracle strategy space** contains the full strategy space of the **Defender Oracle**, then the game value of the **Compact Layer** will always upper bound the true game value.

### 6. REDUCED LAYER

The Reduced Layer uses the same restricted strategy space for the attacker as the Compact Layer. However, the defender uses the original, unrestricted strategy space to allocate resources. While the reduced layer is more difficult to solve than the compact layer, it allows us to iteratively tighten the upper bounds and avoid more computation in the Optimal Layer. The evaluation of teams in this layer follows Algorithm 5. We additionally reduce the computation effort spent in this layer by warm starting the attacker’s mixed strategy with the solution from the compact layer.

**Algorithm 5: ReducedLayer**

1. Initialize mixed strategy \( a \leftarrow \text{CompactLayer}(A^c) \)
2. do:
3. \( X_i^c \leftarrow \text{DefenderOracle}(a) \)
4. \( X^c \leftarrow X^c \cup \{X_i^c\} \)
5. \( (x, a) \leftarrow \text{MiniMax}(X^c, A^c) \)
6. until convergence \( U_d(X_i^c, a) - U_d(x, a) \leq \epsilon \)

**Figure 4:** Compact Graph

![Compact Graph](image)

### 7. EVALUATION

We present four sets of experimental results: (1) We evaluate the scalability and runtime performance of FORTIFY on several classes of random graphs. We benchmark with a sequential search which sequentially evaluates enumerated teams with cost saturating the budget. (2) We also evaluate the impact of the initial compact layer on the runtime by comparing the runtimes of FORTIFY with and without the compact layer. (3) We investigate the benefit of optimizing team composition as well as the diversity of optimal teams and (4) we demonstrate that FORTIFY can scale up to the real world by testing performance on a case study of Madagascar using real graph data. All values are averaged over 20 trials. The experiments were run on a Linux cluster with HP-SL250, 2.4 GHz, dual-processor machines. We use the following graphs:

1. **Grid graphs** Labeled \( G_{w,h,s,t} \) consist of a grid with width \( w \), height \( h \), sources \( s \), targets \( t \) and nearest neighbor connections between nodes. We define start and end points for the attacker, with sources located at one end and targets at another.

2. **Geometric graphs** provide a good approximation of real road networks [3] allowing us to model the networks of villages and rivers in forest regions. \( n \) nodes are distributed randomly in a plane and are connected based on their distance which determines the density \( d \) of the graph. We label them \( R_{n,s,t,d} \).

#### 7.1 Scalability

We first evaluate the performance of FORTIFY using two sets of resource types, and target values of 50. Each set contains 4 resource types, with varying costs of \( b=5,6,7,8 \). The first set of resource types \( A^1 \) have varied path coverages \( L^1=\{1,2,3,4\} \) and constant detection probability \( P^1=\{1,1,1,1\} \) while the second set \( A^2 \) has constant path coverage \( L^2=\{2,2,2,2\} \) and varied detection probabilities \( P^2=\{0.5,0.6,0.7,0.8\} \). Experiments that did not terminate in 3 hrs were cutoff and are shown as missing bars. Figure 5 show the runtimes for our algorithms run on both graph types for

---

**Algorithm 4: Compact Layer**

1. \( G^c \leftarrow \text{CompactGraph}(G(N,E)) \)
2. Initialize mixed strategy \( a \leftarrow \text{Uniform}(A^c) \)
3. do:
4. \( X_i^c \leftarrow \text{CompactDefenderOracle}(a) \)
5. \( X^c \leftarrow X^c \cup \{X_i^c\} \)
6. \( (x, a) \leftarrow \text{MiniMax}(X^c, A^c) \)
7. until convergence \( U_d(X_i^c, a) - U_d(x, a) \leq \epsilon \)
8. return \( U_d(x, a) \)
7.3 Real World: Madagascar Protected Forests

We demonstrate the ability of FORTIFY to scale up to real world domains, evaluating the performance on a network constructed from GIS data of at-risk forest areas in Madagascar. We present the following model which was built working closely with domain experts from NGOs.

**Graph:** Figure 1 shows the road and river networks used by the patrolling officers, as well as the known routes taken by groups of illegal loggers. We used this to build the nodes and edges of our network. Edges correspond to distances of 7-10 km. 10 target locations were chosen by clustering prominent forest areas. 11 villages in the surrounding area were chosen as sources. Several domain experts identified the risk level and level of attractiveness for logging, based on the size of the forest, the ease of access and the value of the trees. Using this information we assigned values ranging from 100 to 300 to each of the targets.

**Resources:** Communal policemen and local volunteers conduct patrols in the forest. A typical patrol covers 20 km in a day and patroller can conduct two types of patrols, a short patrol covering 2 edges and a long patrol covering 3 edges. Based on expert input, we assign the detection probability for communal police as 0.9 for short patrols and 0.8 for long patrols; and for volunteers, 0.7 for short patrols and 0.6 for long patrols. The lower probabilities for volunteers are because they must call backup for interdiction, which may allow the adversary to escape. Thus, in total we have 4 resource types available \( L = \{2,3,2,3\} \), \( P = \{0.7,0.6,0.9,0.8\} \).

The costs are proportional to the salaries patrollers receive for a day of patrolling \( b = \{5,5,8,8\} \).

**Experiment:** The runtime experiments are shown in Table 2 for increasing budgets. Data is averaged over 20 runs. FORTIFY can scale up to real world networks, able to handle both the large graph size and number of source and target nodes, even for large budgets. The value of performing this optimization is shown in Figure 7 with the solution quality (game value) on the y-axis and budget on the x-axis, where we compare the optimal game value to the average value achieved by randomly generated teams.

8. CONCLUSION AND RELATED WORK

We study a fundamentally new problem in Security Games: SORT. This new problem addresses the environmental pro-

![Figure 5: Runtime scalability comparing FORTIFY against Sequential and No-Compact method. (a) \( \Lambda^1 \) teams on a \( G_{5,20,5,5} \) graph. (b) \( \Lambda^2 \) teams on a \( G_{4,4,4,4} \) graph. (c) \( \Lambda^1 \) teams on a \( R_{70,5,5,0.1} \) graph. (d) \( \Lambda^2 \) teams on a \( R_{25,4,4,0.1} \) graph.](image)

![Figure 6: Team optimization comparison. Teams have 6 resource types, and vary both edge coverage \( L = \{2,5,3,3,6\} \), and detection probability \( P = \{0.7,0.9,0.7,0.6,0.6,0.6\} \) with costs \( b = \{5,8,10,5,8,10\} \).](image)
tection challenge of optimal investment and deployment of security resource teams. We use the rising threat of illegal logging in Madagascar as a motivating domain where we must work with a limited budget to coordinate and deploy such teams. To address this problem, we develop FORTIFY, a scalable solution addressing both aspects of the SORT problem, with the ability to both model and solve real world problem instances. FORTIFY provides a valuable tool for environmental protection agencies.

We have already discussed the shortcomings of related work in security games earlier. In addition, there is significant research in team formation in multiagent systems, e.g., in network configuration [5], board gameplay [14] fantasy football [11] and multi-objective coalition [1]). However, that work fails to security resource allocation at the tactical level.

9. ACKNOWLEDGEMENTS

We would like to thank Dr. Jonah Ratsimbazafy. This research was supported by MURI Grant W911NF-11-1-0332.

REFERENCES


