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

Sara Mc Carthy, Milind Tambe
University of Southern California

Christopher Kiekintveld
University of Texas at El Paso

Meredith L. Gore, Alex Killion
Michigan State University

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*.

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 (WWF 2015). As a result, improving the protection of forests is of great concern for many countries (Dhital, Vololomboahangy, and Khasa 2015; Thomas F. Allnutt 2013).

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 (Tambe 2011; Korzhyk, Conitzer, and Parr 2010). 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) (Fang, Stone, and Tambe 2015; Johnson et al. 2012; Nguyen et al. 2015), and *network security games* (NSG) (i.e., interdiction of adversaries on transportation networks) (Jain et al. 2011). 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 heterogenous 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.*

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

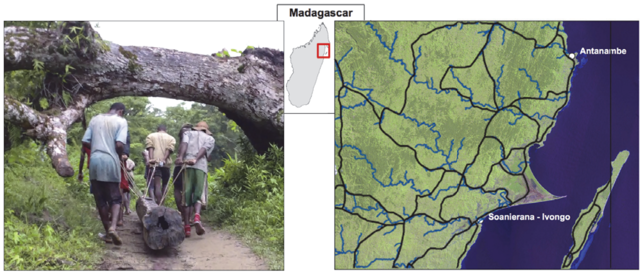


Figure 1: Illegal logging in progress in at risk area of Madagascar, images provided by our partnering NGOs working in the area.

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.

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.

Previously 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 (Jain, Conitzer, and Tambe 2013; Okamoto, Hazon, and Sycara 2012). 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_k 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

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_k	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
$U_d(X_i, \mathbf{a})$	Defender utility playing X_i against **a**
$U_a(\mathbf{x}, A_j)$	Attacker utility playing A_j against **x**

Table 1: Notation and Game Description

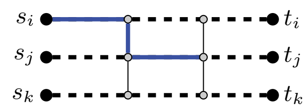


Figure 2: Pure strategies for the Defender (bold line) and Attacker (dashed line going from s to t).

requirement in the game. The allocation of one resource of size $L_k = 2$ 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.

$$P(e, X_i) = 1 - \prod_1^K (1 - P_k)^{m_{k,e}} \tag{1}$$

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.

$$P(X_i, A_j) = 1 - \prod_{e \in A_j} (1 - P(e, X_i)) \tag{2}$$

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 minimax strategy. The game value is denoted $F(\lambda)$, and is function of a team of resources λ 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.

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

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.[1]

[1]Supplemental material and proofs are available at

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 λ. 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 Λ, 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.

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.

http://teamcore.usc.edu/papers/2016/SortAppendix.pdf

Algorithm 1: FORTIFY(B,R)

1: $\Lambda \leftarrow$ getTeams(B,R), $\Lambda^c = \emptyset, \Lambda^r = \emptyset, \Lambda* = \emptyset$
2: **for each** $\lambda \in \Lambda$:
3: $\quad\lambda$.value $\leftarrow$ Compact Layer(λ)
4: $\quad\Lambda^c \leftarrow \Lambda^c \cup \{\lambda\}$
5: **repeat:**
6: $\quad\lambda_{max} \leftarrow \arg\max_{\lambda.\text{value}}(\Lambda^c, \Lambda^r, \Lambda*)$
7: $\quad$if ($\lambda_{max} \in \Lambda*$) **return** λ_{max}
8: $\quad$else: λ.value $\leftarrow$ NextLayer(λ_{max})
9: $\qquad\Lambda^{NextLayer} \leftarrow \Lambda^{NextLayer} \cup \{\lambda_{max}\}$

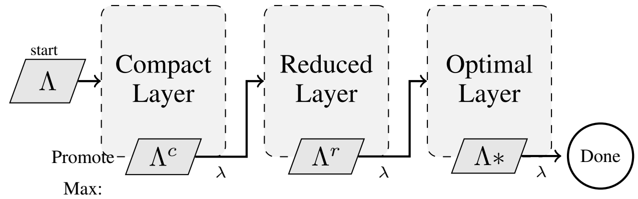


Figure 3: Flowchart for the FORTIFY. Λ is the initial set of teams. Λ^c, Λ^r and $\Lambda*$ are the sets of teams which have passed through the compact, reduced and optimal layers. After all teams pass through the compact layer one team (with max value) is promoted at each step.

Algorithm 2: Optimal Layer(λ)

1: Initialize **X, A**
2: **do:**
3: $\quad$(U_d^*,**x, a**) $\leftarrow$ MiniMax(**X, A**)
4: $\quad$X $\leftarrow$ DefenderOracle(**a**)
5: $\quad$**X** $\leftarrow$ **X** $\cup$ { X }
6: $\quad A_j \leftarrow$ LinearAttackerOracle(**x**)
7: $\quad$if U_a(**x**, A_j) - U_a(**x, a**) $\leq \epsilon$ then
8: $\qquad A_j \leftarrow$ AttackerOracle(x)
9: $\quad$**A** $\leftarrow$ **A** $\cup$ { A_j }
10: **until** convergence then **return** (U_d^*,**x**)

Optimal Layer

We first introduce the optimal layer of FORTIFY in order to explain how the full NSG is solved. This step is computationally expensive as there are an exponential number of attacker and defender strategies and explicitly enumerating them all in computer memory is infeasible. Incremental strategy generation addresses the first challenge, allowing us to obtain the optimal defender mixed strategy without enumerating all pure strategies (McMahan, Gordon, and Blum 2003). This approach decomposes the problem into (1) master MiniMax linear program (LP) and (2) oracles for the defender and attacker which incrementally generate pure strategies to add to the current strategy space via a separate optimization problem. The master LP computes a solution restricted to the set of pure strategies generated by the oracles. The steps are shown in Algorithm 2. The formulation for the MiniMax LP is standard, but we provide new formulations for both oracles. The key novelty in our work is that the model complexities imply that we have a non-linear optimization problem to solve in both oracles; we address this by constraining the variables to be binary valued, and we take advantage of efficient constraint programming methods in commercial tools like CPLEX.

Minimax: The game value is computed (line 3) by solving for the minimax strategy using a LP formulation (4). The inputs are the current set of attacker and defender pure strategies, **A** and **X** and the outputs are the game utility U_d^*, and mixed strategies, **x** and **a**. $U_d(\mathbf{x}, A_j)$ is the defender utility playing **x** against the pure strategy A_j.

$$\max_{U_d^*, \mathbf{x}} \quad U_d^* \quad \text{s.t.} \quad U_d^* \leq U_d(\mathbf{x}, A_j) \quad \forall j = 1 \ldots |A| \tag{4}$$

Defender Oracle: The defender oracle returns the best response strategy X_i 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.

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

A pure strategy implies a single allocation of the given team's resources. Resources are allocated by setting the binary decision variables $\lambda^k_{m,e} \in \{0,1\}$ 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 $\sum_e \lambda^k_{m,e} = L_k$ 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. $\lambda^k_{n,m} \in \{0,1\}$ and equals 1 if a node n is the source or target of any allocated edge $\lambda^k_{e,m} = 1$.

$$\lambda^k_{m,e} \leq \sum_{e1\in in(n_s)} \lambda^k_{m,e1} + \sum_{e2 \in out(n_t)} \lambda^k_{m,e2} \quad \substack{n_s \leftarrow \text{source(e)} \\ n_t \leftarrow \text{target(e)}} \quad \text{if } L_k \geq 1 \tag{6}$$

$$\lambda^k_{m,e} \leq \lambda^k_{m,n} \quad \text{s.t. } n \leftarrow \substack{\text{source(e)} \\ \vee\ \text{target(e)}} \quad \sum_n \lambda^k_{m,n} = L_k + 1 \tag{7}$$

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 $e \in A_j$.

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

$$\sum_{e\in out(s)} \gamma_e = 1; \quad \sum_{e \in in(t*)} \gamma_e = 1; \quad \sum_{e\in in(n)} \gamma_e = \sum_{e \in out(n)} \gamma_e \quad \substack{n \neq \text{source} \\ n \neq \text{target}} \tag{9}$$

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, $P(X_i, A_j) = \sum_{e \in A_j} P(e, X_i)$ so we can write the oracle as an LP. (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.) 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.

Algorithm 3: CompactGraph(G(N,E))

1: for each $s_i \in N$, $t_j \in N$:
2: $\{E_j\} \leftarrow$ mincut(s_i,t_j)
3: for each $e \in E_j$:
4: $A^c \leftarrow$ShortestPath(s_i, e, t_j)
5: $\mathbf{A^c} \leftarrow \mathbf{A^c} \cup \{A^c\}$
6: $N^c \leftarrow N^c \cup$ newNode(A^c)
7: for each $A^c_i \in \mathbf{A^c}$, $A^c_j \in \mathbf{A^c}$:
8: $w_{i,j} \leftarrow D(i,j)$
9: $G^c \leftarrow$ newEdge($i, j, w_{i,j}$)
10: return G^c

Compact Layer

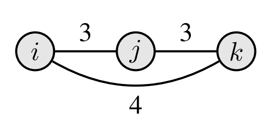
The compact layer uses an *abstract representation* of the game model which reduces the problem in two ways: (1) the attacker is restricted to using *only a subset* of possible paths, (2) the defender chooses to allocate resources *directly to attacker paths* rather than edges in the graph.

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_i, A_j and A_k in Figure 2 are among several calculated from the min-cut of the graph. These three paths become nodes i, j, and k respectively, in part of 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 to cover both corresponding paths. 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 resource of size 4. If the defender has a resource of size 3, she can only cover paths (i-j) or (j-k).

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

Figure 4: Compact Graph



Algorithm 4: Compact Layer

1: $G^c \leftarrow$ CompactGraph(G(N,E))
2: Initialize mixed strategy **a** $\leftarrow$ Uniform(A^c)
3: **do:**
4: $\quad X_i^c \leftarrow$ CompactDefenderOracle(**a**)
5: $\quad \mathbf{X^c} \leftarrow \mathbf{X^c} \cup \{X_i^c\}$
6: $\quad$(**x, a**) $\leftarrow$ MiniMax($\mathbf{X^c}, \mathbf{A^c}$)
7: **until** convergence $U_d(X_i^c$, **a**) - U_d(**x, a**) $\leq \epsilon$
8: **return** U_d(**x, a**)

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 10 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 $(1-(1-P_k)^{L_k})$.The defender chooses to allocate the m^{th} resource of type k to a node n_j corresponding to attacker path A_j by setting the decision variables $\eta_{j,m}^k \in \{0,1\}$.

$$P(X_i, A_j) = \sum P_k^c \eta_{j,m}^k \qquad D(i,j)\eta_{i,m}^k \eta_{j,m}^k \leq L_k \tag{10}$$

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.*

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

Algorithm 5: ReducedLayer

Initialize mixed strategy **a** $\leftarrow$ CompactLayer(A^c)
do:
$\quad X_i^c \leftarrow$ DefenderOracle(a)
$\quad \mathbf{X^c} \leftarrow \mathbf{X^c} \cup \{X_i^c\}$
$\quad$(**x, a**) $\leftarrow$ MiniMax($\mathbf{X^c}, \mathbf{A^c}$)
until convergence $U_d(X_i^c$, **a**) - U_d(**x, a**) $\leq \epsilon$
return U_d(**x, a**)

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.

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 (Eppstein and Goodrich 2008) 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 $R_{n,s,t,d}$.

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 Λ^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 Λ^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 both Λ^1 and Λ^2 teams. The budget varies on the x-axis and the runtime is shown on the y-axis in log scale. FORTIFY consistently outperform the sequential method; on both the grid and geometric graphs. FORTIFY

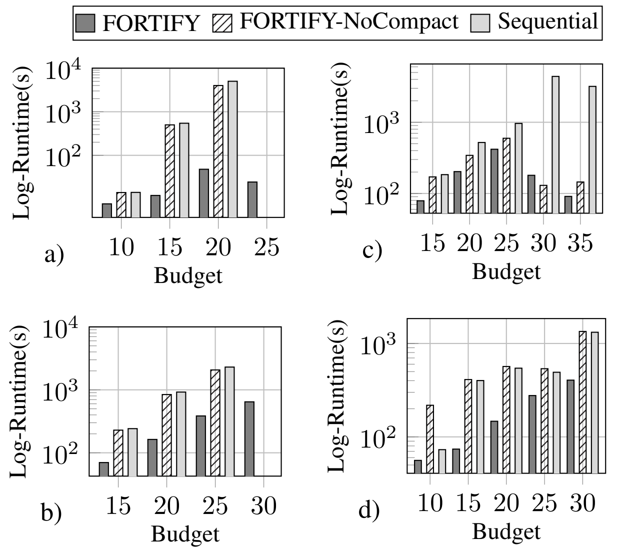


Figure 5: Runtime scalability comparing FORTIFY against Sequential and No-Compact method. (a) Λ^1 teams on a $G_{5,20,5,5}$ graph. (b) Λ^2 teams on a $G_{4,4,4,4}$ graph. (c) Λ^1 teams on a $R_{70,5,5,0.1}$ graph. (d) Λ^2 teams on a $R_{25,4,4,0.1}$ graph.

performs particularly well on the grid graphs, and scaling past budgets of 25 while all instances of the sequential search were cut off. We observe a peak in the runtime for teams with perfect detection probability in 5 (a) and (C) around a budget of 20-25, which is due to the deployment vs saturation phenomenon which occurs in these types of network models (Jain, Leyton-Brown, and Tambe 2012).

Removing the Compact Layer: We also compare the performance of FORTIFY with and without the compact layer in Figure 5. It is apparent that this layer is crucial to the performance of the algorithm, particularly for the grid graphs in parts (a-b) as FORTIFY without the compact layer performs almost as poorly as the sequential method. In fact, removing the compact layer can cause FORTIFY to perform worse than the sequential method for small budgets due to the overhead required for the approximation.

Team Composition

We demonstrate the value of optimizing over team composition by looking at the loss in game value incurred by playing a uniform random team which saturates the budget.

Games are played on $G_{4,4,4,4}$ and $R_{25,4,4,0.1}$ graphs with target values of 50. The results are shown in Figure 6 with budget on the x-axis and game value on the y-axis. As expected, the game value decreases with budget as we form larger teams, however the relative benefit increases as well, with almost a 300% loss in solution quality at budgets of 25 without our team optimization algorithm. This is due to the increase in the space of possible teams which can be formed, making it more likely to form a suboptimal team.

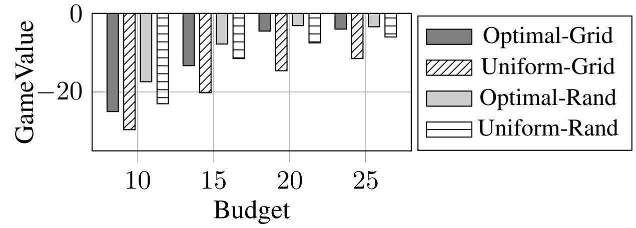


Figure 6: Team optimization comparison. Teams have 6 resource types, and vary both edge coverage $L = \{2, 2, 5, 3, 3, 6\}$, and detection probability $P = \{0.7, 0.9, 0.7, 0.6, 0.6, 0.6\}$ with costs $b = \{5, 8, 10, 5, 8, 10\}$.

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.

Conclusion and Related Work

We study a fundamentally new problem in Security Games–SORT. This new problem addresses the environmental protection challenge of optimal investment and deployment of

B	Runtime(s)	GV
10	203	-266
15	388	-256
20	653	-239
25	1308	-230
30	1742	-220
35	2504	-216
40	3675	-204

Table 2: Runtime on Madagascar Graph

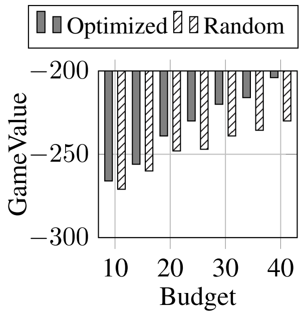


Figure 7: Team optimization on Madagascar Graph

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 (Gaston and desJardins 2005), board gameplay (Obata et al. 2011) fantasy football (Matthews, Ramchurn, and Chalkiadakis 2012) and multi-objective coalition (Cho et al. 2013)). However, that work fails to security resource allocation at the tactical level.

Acknowledgements

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

References

Cho, J.-H.; Chen, I.-R.; Wang, Y.; Chan, K. S.; and Swami, A. 2013. Multi-objective optimization for trustworthy tactical networks: A survey and insights. Technical report, DTIC Document.

Dhital, N.; Vololomboahangy, R. R.; and Khasa, D. P. 2015. Issues and challenges of forest governance in madagascar. *Canadian Journal of Development Studies / Revue canadienne d'tudes du dveloppement* 36(1):38–56.

Eppstein, D., and Goodrich, M. T. 2008. Studying (non-planar) road networks through an algorithmic lens. In *Proceedings of the 16th ACM SIGSPATIAL international conference on Advances in geographic information systems*, 16. ACM.

Fang, F.; Stone, P.; and Tambe, M. 2015. When security games go green: Designing defender strategies to prevent poaching and illegal fishing. In *International Joint Conference on Artificial Intelligence (IJCAI)*.

Gaston, M. E., and desJardins, M. 2005. Agent-organized networks for dynamic team formation. In *AAMAS*, 230–237. ACM.

Jain, M.; Korzhyk, D.; Vaněk, O.; Conitzer, V.; Pěchouček, M.; and Tambe, M. 2011. A double oracle algorithm for zero-sum security games on graphs. In *AAMAS*, 327–334.

Jain, M.; Conitzer, V.; and Tambe, M. 2013. Security scheduling for real-world networks. In *AAMAS*, 215–222.

Jain, M.; Leyton-Brown, K.; and Tambe, M. 2012. The deployment-to-saturation ratio in security games. In *Conference on Artificial Intelligence (AAAI)*.

Johnson, M. P.; Fang, F.; ; and Tambe, M. 2012. Patrol strategies to maximize pristine forest area. In *Conference on Artificial Intelligence (AAAI)*.

Korzhyk, D.; Conitzer, V.; and Parr, R. 2010. Complexity of computing optimal stackelberg strategies in security resource allocation games. In *AAAI*.

Matthews, T.; Ramchurn, S. D.; and Chalkiadakis, G. 2012. Competing with humans at Fantasy Football: Team formation in large partially-observable domains. In *Proceedings of the 26th Conference of the Associations for the Advancement for Artificial Intelligence*, 1394–1400.

McMahan, H. B.; Gordon, G. J.; and Blum, A. 2003. Planning in the presence of cost functions controlled by an adversary. In *In Proceedings of the Twentieth International Conference on Machine Learning*.

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.; and Rwetsiba, A. 2015. Making the most of our regrets: Regret-based solutions to handle payoff uncertainty and elicitation in green security games. In *Conference on Decision and Game Theory for Security*.

Obata, T.; Sugiyama, T.; Hoki, K.; and Ito, T. 2011. Consultation algorithm for Computer Shogi: Move decisions by majority. In *Computer and Games'10*, volume 6515 of *Lecture Notes in Computer Science*, 156–165. Springer.

Okamoto, S.; Hazon, N.; and Sycara, K. 2012. Solving non-zero sum multiagent network flow security games with attack costs. In *AAMAS*, 879–888.

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

Thomas F. Allnutt, Gregory P. Asner, C. D. G. G. V. N. P. 2013. Mapping recent deforestation and forest disturbance in northeastern madagascar. *Tropical Conservation Science* 6(1):1–15.

WWF. 2015. http://wwf.panda.org/about_our_earth/deforestation/deforestation_causes/illegal_logging.