Approximating the Allocation of Heterogeneous Resources to Prevent Illegal Logging

No Author Given

No Institute Given

Abstract. In the present work, we study the problem of optimizing non-homogeneus resources for the defense of forests against illegal logging, a serious issue that is affecting more and more developing countries. In fact, the protection of natural environments and wildlife is one of the most important challenges of our century. Unfortunately, in these cases, there is only a small number of resources available to patrol and defend these areas w.r.t. their vastness. Starting from a model already presented in the literature, we first introduce some properties that limit the effectiveness of a direct exact approach. Thus, we focus on finding an approximate solution, showing that approximating our problem is APX-complete. Nevertheless, we provide a very fast and performing algorithm based on some simple but effective heuristics. We evaluate our methods on synthetic examples, as well as a real-world case study, using data from our on-going collaboration in Madagascar.

1 Introduction

During the last decades, illegal logging has become one of the major issues both for global forest policy [23] and for several developing countries. According to [25], 80% of logging in the Amazon forests in Brazil violated the government controls, [19] reports that illegal logging contributes to more than 50% of tropical deforestation in Central Africa, the Amazon Basin and South East Asia and [27] states that in the Democratic Republic of Congo illegal logging happens at a rate of 65%, and this value increases up to 80% and 85% for Perú and Myanmar, respectively. This has a strong impact also on the US industry: a study of the American Forest and Paper Association estimated that illegal logging costs the U.S. forest products industry $1 billion annually in lost export opportunities [1]. Thus, developing an effective protection of the forests in these areas is a very serious issue for many countries [2, 8]. Such protection should be also affordable: these countries have a very limited budget they can spend to solve this problem, making crucial allocating at best the available resources. The present work focus on the deployment of non-homogeneus resources to interdict the traversal of illegal loggers on a network of roads and rivers around the forest areas. We consider non-homogeneus resources since there are different organizations that may be involved in the defensive patrols, e.g., local volunteers, police units, NGO personnel, each differing in their detection skills, both individual and jointly with other patrollers, and costs of deployment. Thus, given some budget, we can build a very large number of teams and, for each of them, we have even more allocation strategies, all with varying effectiveness. Our goal is designing scalable and experimentally reliable algortithms to both find the best team of resources and compute its best allocation in order to maximize the protection of forests against a malicious attacker. To accomplish this task, we exploit game theoretical tools and multiagent systems techniques.

Security Games represent a successful application of non-cooperative game theory in the real world, e.g., [], and in particular for resource allocation [17, 24]. Finding such best allocation is currently one of the most challenging problems in Artificial Intelligence [12]. Customarily, a security game is a 2-player game between a *Defender* $\mathcal{D}$ and an *Attacker* $\mathcal{A}$, which takes place under a Stackelberg (a.k.a. leaderfollower) paradigm [26], where the Defender (leader) commits to a strategy and the Attacker (follower) first observes such commitment and then best responds to it. As discussed in the seminal work [7], finding a leader-follower equilibrium is computationally tractable in games with one follower and complete information, while it becomes hard in Bayesian games with different types of Attacker. Despite their great effectiveness to solve real problems, the models presented in the literature are quite simple and only few works introduce interaction between the actions played by the Defender and the Attacker, e.g., in [22] the Attacker observes the actions of the Defender while in [4–6] the Defender is supported by an alarm system triggering alarm signals when a target is attacked.

We now restrict our attention to Network Security Games (NSG), which constitute the class of games the one studied in this work belongs to, where the Defender must protect some targets placed in an environment usually represented as a graph. [14] proposes a double-oracle approach, i.e., interdiction of adversaries on transportation networks, while [15, 10, 20] deals with the protection of forests, fish and wildlife. Even thoughthese works deal, once

more, with real-life problems, they present two main limitations. First, they only consider the deployment of an already given team of resources, without taking into account the strategic question of the composition of the best team. Second, they only focus on homogeneous resources, i.e., resources with the same cost and skills (see [13, 21]). [18] proposes an exact algorithm to deal with these limitations. The main problem with this approach is scalability: the approach presented in such work, namely FORTIFY, is exact and, being the problem NP-hard, has an exponential computational cost. In fact, FORTIFY can solve only instances with few resources and a relatively low budget to build the team employed by $\mathcal{D}$, being the space of possible teams and their allocations quiet restricted. Thus, the computational issue is still open and very challenging.

Original contributions In order to address scalability and be able to face more challenging and complex real-world scenarios, we provide the following original contributions. First, we introduce the concept of dominance among teams and a new utility function, which allows us to effectively compare different teams. Then, we provide some results about the relations between resources and teams, and among teams themselves. Unfortunately, such results are mostly negative, showing that we cannot exploit the knowledge about the best allocation in the environment for one team to infer information on another one. Such results are not restricted to our domain, but can be applied to any team formation setting where the utility function is submodular (a very common assumption for such problems). Starting from these negative results, we are naturally driven to investigate approximation algorithms, to find good solution in a faster way. Unfortunately, computing an approximate solution to our problem is APX-hard. Nevertheless, we provide some approximation algorithms based on an incremental greedy approach. Finally, we test such algorithms both on synthetic instances and on a real-world instance obtained from joint work with NGOs engaged in forest protection in Madagascar. We show that, despite their simplicity, such algorithms provide a very high utility ratio and are much faster than the exact approach already presented in the literature.

2 Starting point

We borrow the description of the model from [18], where the interactions among the different groups of resources are represented both at strategic and tactical levels. After describing the model, we also sketch the current exact approach to solve the problem.

2.1 Simultaneous Optimization of Resource Teams and Tactics (SORT)

As customary in the literature, this is a 2-player game, where a Defender $\mathcal{D}$ faces an Attacker $\mathcal{A}$. The environment is modeled as a graph $G = (V, E)$, where the nodes correspond to different areas and the edges represent the connections among them. There are three types of nodes: source nodes $s \in S \subset V$, target nodes $t \in T \subset V$ and intermediate nodes. Source nodes s_i are the starting points for the illegal loggers, e.g., villages, while target nodes t_j are areas with valuable trees, each characterized by a value $\tau(t_j)$ that depends on the real domain. The goal of the Attacker $\mathcal{A}$ is traversing the graph from a source to a target without being detected by the Defender $\mathcal{D}$. Thus, the pure strategies A_i of $\mathcal{A}$ are all the possible paths from any source s_i to any target t_j. On the other side, the $\mathcal{D}$ wants to intercept $\mathcal{A}$ while it is moving along the edges. The Defender has a budget B she can spend to build a team of resources that can be chosen from a pool. There are $K = \{1, 2, \ldots, k\}$ types of resources, each characterized by a cost c_k, a number of edges L_k it can cover and a detection probability P_k of identifying $\mathcal{A}$. Indeed, as often happens in reality, the resources may not be able to perfectly detect $\mathcal{A}$. Thus, given that some team λ has been built by $\mathcal{D}$, her the pure strategies $X_j(\lambda)$ are all the possible allocations of the resources of team λ to the edges of the graph.

The probability of intecepting $\mathcal{A}$ on edge e is equal to:

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

where $m_{k,e}$ is the number of resources of type k (a.k.a. its multiplicity) that are covering e. The protection that $X_i(\lambda)$ can ensure against A_j is equal to the following probability:

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

The game is zero-sum: if $\mathcal{D}$ can detect $\mathcal{A}$, both players get a utility equal to 0, while if the Attacker can complete its attack, it will get the value $\tau(t_i)$ of target t_i she attacked and the Defender will lose the same amount, getting a

utility equal to $-\tau(t_i)$. Being the game value a function of some team λ, we denote it as $V(\lambda)$. In Table 1, we report all the symbols we adopt throughout the paper.

Symbol	Meaning
$\mathcal{D}$	Defender
$\mathcal{A}$	Attacker
$G = (V, E)$	Graph modeling the environment
t_i	i-th target
$\tau(t_i)$	Value of target t_i
B	Budget available to $\mathcal{D}$ to build a team
r_i	i-th resource
L_i	Number of edges covered by r_i
P_i	Detection probability of r_i
c_i	Cost of r_i
K	Set of types of possible resources
A_j	$\mathcal{A}$'s j-th pure strategy
$\mathbf{a}$	$\mathcal{A}$'s mixed strategy
λ	Generic team of resources
$X_i(\lambda)$	i-th $\mathcal{D}$'s pure strategy given team λ
$\mathbf{x}$	$\mathcal{D}$'s mixed strategy

Table 1. Symbols table.

2.2 FORTIFY

Solving SORT requires to build a candidate team and then computing its value, obtained by best allocating its resources on the edges of the graph. Such value corresponds to the utility for the Defender. Unfortunately, solving exactly this problem requires exponential time, being it NP-hard. Nevertheless, an exact algorithm, FORTIFY (Forming Optimal Response Teams for Forest safetY), has been proposed in [18]: such algorithm integrates the analysis of the strategic and tactical aspects of the problem to search the space of teams efficiently. First, it enumerates all teams λ that maximally saturate the budget B. Then, it uses a three-layers 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 $V(\lambda)$. Finally, a team is discarded if its value is lower than some bound. In other words, FORTIFY adopts fast methods to quickly evaluate upper bounds on the utilities for specific teams and exploits such bounds to select the most promising team to evaluate more in detail, iteratively tightening the bounds as the search progresses, until the optimal team is identified. The main drawback of this approach is the limited scalability. In fact, either when the numbers of resources or the budget increase, on one side the number of teams to evaluate increase dramatically, on the other side the bounds are not tighten enough, meaning that a lot of teams will reach the last layer, the one that solves the actual game, which is the most expensive in terms of computational cost.

3 (Not) Extending FORTIFY

The Defender's payoffs are always non-positive in our domain since $\mathcal{D}$ gets a payoff of $-\tau(t_i)$ when the Attacker successfully attacks target t_i, and 0 otherwise. To facilitate our analysis, as done in [13], we define a non-negative normalized utility function f_d for the Defender. Given a team λ,

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

$$f_d(X_i(\lambda)) = U_d(X_i(\lambda), \mathbf{a}) - U_d(\varnothing, \mathbf{a})$$

More specifically, f_d gives the added benefit of the Defender allocation $X_i(\lambda)$ over the Defender not protecting any edges. We observe that f_d is submodular in the resources, i.e., $f_d(X^*(\lambda)) + f(X^*(r)) \geqslant f(X^*(\lambda \cup r))$, where λ

is some team, r some resource and $X^*(\cdot)$ the best allocation for some team. In the following, to simplify the notation, we will just write $f_d(\lambda)$, assuming we are considering the value of λ when its resources are allocated at best.

3.1 The hardness of team/resource domination

The first step of FORTIFY is the enumeration of all the teams that saturate the budget: the main weakness here is the high number of teams that reach the last layer before the best one is found. Thus, if we could safely exclude some teams from the candidate list *before* evaluating them, then we could also reduce the total time required by FORTIFY. Thus, given the list of teams, we investigate dominance relations among similar teams. For our purposes, two teams λ, λ' are called *similar* if they differ for at most $d(\lambda, \lambda')$ resources. Unfortunately, even with $d(\lambda, \lambda') = 1$, we obtain the following negative results.

Theorem 1. *Let r_1, r_2 be two resources of type $1, 2 \in K$, respectively. If $f_d(\{r_1\}) \geqslant f_d(\{r_2\})$, then, for any team λ, $f_d(\lambda \cup \{r_1\}) \geqslant \frac{1}{2} f_d(\lambda \cup \{r_2\})$.*

Proof. By the submodularity of f_d, we know that:

$$f_d(\lambda) + f_d(\{r_i\}) \geqslant f_d(\lambda \cup \{r_i\}), i = 1, 2$$

Moreover, the following inequalities hold:

1. $f(\lambda \cup \{r_1\}) \geqslant f(\lambda)$
2. $f(\lambda \cup \{r_1\}) \geqslant f(\{r_1\})$
3. $f(\lambda) + f(\{r_1\}) \geqslant f(\lambda) + f(\{r_2\})$
4. $f(\lambda) + f(\{r_2\}) \geqslant f(\lambda \cup \{r_2\})$

From 1. and 2. we get:

$$f(\lambda \cup \{r_1\}) + f(\lambda \cup \{r_1\}) \geqslant f(\lambda) + f(\{r_1\})$$

Combining the above result with 3. and 4., we can write:

$$f(T \cup \{r_1\}) + f(\lambda \cup \{r_1\}) \geqslant f(\lambda \cup \{r_2\})$$

and, finally:

$$f(\lambda \cup \{r_1\}) \geqslant \frac{1}{2} f(\lambda \cup \{r_2\})$$

□

The following theorem also holds.

Theorem 2. *Let r_1, r_2 be two resources of types $1, 2 \in K$, respectively. If $f(\{r_1\}) \geqslant f(\{r_2\})$, then we cannot say if for any team λ, $f(\lambda \cup \{r_1\}) \geqslant f(\lambda \cup \{r_2\})$.*

Proof sketch. We prove this theorem providing the following counterexample. Let us consider a 4×4 grid graph, with 4 sources on one side and 4 targets, on the opposite side, with $\tau(t_i) = 20$ for all t_i. Let r_1, r_2 be two resources of types $1, 2 \in K$ with the following features:

	L_i	P_i
r_1	2	0.9
r_2	4	0.45

and let $\lambda = \{r_1\}$.
Solving exactly the problem for $\{r_1\}$, $\{r_2\}$, $\lambda \cup \{r_1\}$, $\lambda \cup \{r_2\}$, we obtain the following values:

- $f(\{r_1\}) = 4.95$;
- $f(\{r_2\}) = 4.17$;
- $f(\lambda \cup \{r_1\}) = 9.9$;
- $f(\lambda \cup \{r_2\}) = 10.1$.

We observe that even though $f(\{r_1\}) \geqslant f(\{r_2\})$, we have $f(\lambda \cup \{r_1\}) \leqslant f(\lambda \cup \{r_2\})$. This concludes the proof. □

Theorem 1 and Theorem 2 tell us that we cannot exclude from the candidate list some team λ knowing the exact value of another team λ', even if $d(\lambda, \lambda') = 1$. Actually, the implications of such theorems are deeper: we express them in the following corollaries (we omit their proofs since they easily follow from the proof of Theorem 2).

Corollary 1. *Given a team λ, we cannot infer any information about its value taking into account only the features L_k, P_k, c_k of the resources that constitute the team.*

Corollary 2. *Given two teams λ, λ', we cannot say whether or not $f(\lambda) \geqslant f(\lambda')$ taking into account only the features L_k, P_k, c_k of the resources that constitute the team.*

Corollary 3. *Given two teams λ, λ', let C be the set of common resources, i.e., $\lambda \cap \lambda' = C$. Let $f(\lambda \backslash C) \geqslant f(\lambda' \backslash C)$. Then, we cannot say whether or not $f(\lambda) \geqslant f(\lambda')$.*

Corollary 4. *Let r_1, r_2 be two resources of types $1, 2 \in K$, and $r_1 \in \lambda$. Then, we cannot say whether or not $f(\lambda) - f(\{r_1\}) + f(\{r_2\}) \geqslant f(\lambda\{r_1\} \cup \{r_2\})$.*

Corollary 5. *Let λ^*, r^* be, respectively, the team ad the single resource that maximize f_d. Then, we cannot say whether or not $r^* \in \lambda^*$.*

As can be seen from the results stated above, improving the current exact approach is a hard task. This is why we have to turn our attention to new techniques. In general, when there is a problem of team formation, two main approaches can be adopted. The first, employed also by FORTIFY, consists in listing a set of possible candidates among the whole space of solutions and reduce their number until the best team is found. The second approach is incremental: given the set of elements of which the team can be composed of, at each iteration we add one element according to some rules, until the budget constraint is violated. The results listed above give us important directions for both methods.

Let us say that we want to follow the first approach. Then, we know that we cannot exclude any team *a priori* just taking into account the resources it is composed of. Moreover, we cannot discard any team even if we know the real value of another team which differs from it for just one single resource. Similarly, if we remove the common resources from two teams and evaluate just the remaining ones, we cannot state anything about the exact values of the original teams. Although listing all the team and discarding the some of them until the best is found offers a safe way to determine the best team, namely the exact value of the best team has to be better than the upper bounds on the values of the other teams, all these negative results suggest that this approach is computationally very expensive since we cannot remove any team from the list, unless the condition on the upper bound is met. For similar reasons, given r_1, r_2 of types $1, 2 \in K$, we cannot say whether $r_1(r_2)$ is better than $r_2(r_1)$ in any team λ (except for very trivial cases, e.g., if $c_1 = c_2, L_1 = L_2, P_1 \geqslant L_2$, r_2 can be safely discarded).

On the other side, let us say that we opt for an incremental approach. Here, we cannot guarantee that some resources will belong to the best team and, besides adopting an expensive complete approach, e.g., the exact algorithm for the Knapsack problem [3], it is difficult to determine a safe way to state that the best team has been found.

4 Towards an approximation approach

For the reasons stated in the previous section, we are naturally driven to study approximation algorithms for our problem, looking for efficient algorithms, along with theoretical guarantees w.r.t. the quality of the solution they provide.

4.1 The hardness of approximating SORT

Before proposing any algorithm, we tackle the approximation problem studying its complexity.

Theorem 3. *The problem of computing the optimal team, SORT, is APX-complete.*

Proof skecth. In order to prove the APX-completeness, we have to show the APX-hardness and the membership of SORT to APX.

APX-hardness. We observe that approximating our problem corresponds to approximating the value of a submodular function, namely f_d, subject to a budgetary constraint. But this problem is actually a special case of the Maxmimum Coverage Problem with cardinality constraints, which is known to be APX-hard [11]. Thus, also our problem results being APX-hard. □

Membership to APX. As we have seen, approximating our problem corresponds to approximating the value of a submodular function subject to a budgetary constraint. To solve this problem, [16] provides a greedy algorithm with an approximation factor of $1 - \frac{1}{e} \approx 0.63$.

This concludes the proof. □

Starting from these results, we design some heuristics to approximate our solution. Due to the efficiency and the effectiveness shown by the greedy approach in solving several problems close to ours, we resort to it to build our approximation algorithms.

4.2 A Polynomial Algorithm for Team formation guided by Heuristics (PATH)

Given that [16] provides an algorithm with the best approximation factor, the first step would be adopting it. However, we cannot employ such algorithm to solve our problem since it computes the value for all the singletons, couples and triplets and then follows a greedy approach to maximize the submodular function until the budget is violated. But in SORT evaluating all the singletons, couples and triplets requires too much computational effort (remember that computing $V(\lambda)$ is hard) and so we have to look for another method. In the same paper, i.e., [16], another algorithm is proposed: it solves the problems for the singletons (in our case, finding the best allocation for the single resources) and then follows a greedy approach. Such algorithm provides a $1 - \frac{1}{\sqrt{e}} \approx 0.39$[1].

PATH takes as input the graph G, the set of the features of the resources L, P, c and the budget B the Defender can use to build the team. First, it sets team $\lambda = \varnothing$ and initialize to 0 an array of size $|L|$ (Lines 1–2). Then, it assigns a value to each resource r_i according to some heuristic $h(r_i)$ and sort the resources in descending order according to such values, invoking the function `Sort`. This way, we obtain an ordered sequence of resources $R' = \langle r'_1, \ldots, r'_k \rangle$ (Lines 3–5). Starting from $i = 1$, PATH adds to λ as many units as possible of r'_i until the budget constraint is violated. If so, it repeats this procedure for r'_{i+1}. The algorithm continues adding resources to λ this way until $i = |K|$ (Lines 6–9). Finally, PATH evaluates λ calling the `ComputeExactValue` (Line 10).

Algorithm 1 `PATH`(G, L, P, c, B)

```
1: λ = ∅
2: ω_i ← 0, i = 1, ..., |K|
3: for all i ∈ K do
4:     ω(r_i) = h({r_i})
5: R' ←Sort(R, ω)
6: for all r_i ∈ R' do
7:     while B − c_i ⩾ 0 do
8:         λ ← λ ∪ {r_i}
9:         B ← B − c_i
10: V =ComputeExactValue(λ)
11: return V
```

Now, we turn to the core of PATH, namely the heuristic we should employ to sort the resources. If we adopt $h_{v,c}(r_i) = \frac{f(\{r_i\})}{c_i}$ (where $_{v,c}$ denote *value* and *cost*, respectively), then PATH completely resembles the algorithm provided in [16]. Thus, adopting $h_{v,c}$ we know we have a guaranteed lower bound of 0.39 w.r.t. the optimal solution. From this incremental greedy approach, we can define other simple but very fast heuristics that can be inserted in PATH. In particular, we provide three additional heuristics, designed according to two criteria. First, we take into account either the *features* of the resource ($h_f(\cdot)$) or the *value* obtained solving exactly the game with a team composed

[1] Even though the algorithm corresponds to the well-known greedy algorithm for the Knapsack problem, we observe we cannot state the $\frac{1}{2}$ approximation because of the submodularity of function f_d.

only of that resource ($h_v(\cdot)$). Moreover, we can also include the cost of the resource ($h_c(\cdot)$). Adopting such criteria, we design the following heuristics:

- $h_f(r_i) = L_i \cdot P_i$;
- $h_{f,c}(r_i) = \frac{L_i \cdot P_i}{c_i}$;
- $h_v(r_i) = f(\{r_i\})$;
- $h_{v,c}(r_i) = \frac{f(\{r_i\})}{c_i}$[2].

The main drawback in employing such heuristics is that we do not have guarantees on the quality of the solution computed by PATH when adopting them w.r.t. the optimal solution computed by FORTIFY (except for $h_{v,c}$). However, in the next section we show that, despite the lack of explicit theoretical guarantees, the quality of the solution and time required by such heuristics are very good w.r.t. FORTIFY.

5 Experimental evaluations

Here, we show that, despite its simplicity, PATH provides very high quality solutions with a significantly small running time w.r.t. FORTIFY, both in synthetic and real-world instances.

5.1 Testbed

To test PATH, we generate and solve instances that are similar to the ones tested in [18], that we know can be solved also by FORTIFY in a reasonable amount of time. Our testbed includes the following graphs:

- Geometric graphs: they provide a good approximation of real road networks [9], allowing us to model the networks of villages and rivers in forest regions. n nodes, which include some source nodes s and some targets t, are distributed randomly in a plane and are connected based on their distance r, which determines the density of the graph. We label such graphs as $R_{n,s,t,r}$.
- Grid graphs $G_{w,h,s,t}$: they consist of a grid with width w, height h, s source nodes, t targets and nearest neighbor connections between nodes. We also define starting and ending points for $\mathcal{A}$, with sources located at one end of the graph and targets at the other.
- Madagascar graph: this graph is a network built from GIS data of at-risk forest areas in Madagascar.

Algorithms are implemented in Java 6u45 and are executed on a Linux cluster with HP-SL250, 2.4 GHz, dual-processor machines. In the figures, the budget varies on the x-axis while on the y-axis we report either the utility ratios or the time ratios of PATH, which adopts the various heuristics, w.r.t. FORTIFY.

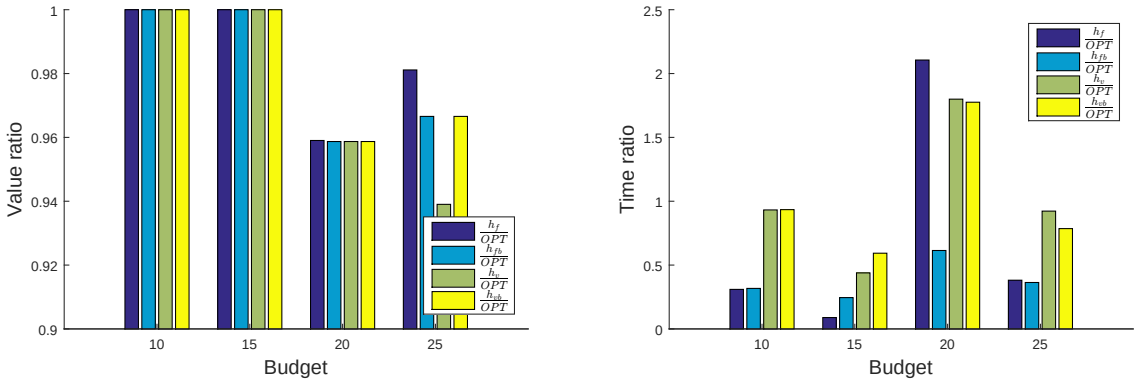
5.2 Synthetic instances

In this section, we focus on synthetic experiments. In particular, we analyze grid graphs $G_{4,4,4,4}$ and geometric graphs $R_{25,4,4,0.1}$. The resources have the following features $L = \{2, 2, 5, 3, 3, 6\}, P = \{0.7, 0.9, 0.7, 0.6, 0.6, 0.6\}, b = \{5, 8, 10, 5, 8, 10\}$ and the Defender can spend budget $B \in \{10, 15, 20, 25\}$. Proceeding this way, on one side we are sure that FORTIFY can solve the problem in a reasonable amount of time, on the other we are not *saturating* the problem, i.e., the budget is not high enough to allow the construction of a team that covers perfectly each target.

Grid graphs. Figure 1(a) shows that for low values of the budget, all the heuristics select the same team, getting an approximation ratio higher than 95% with a budget of 20. If we look at a budget equal to 25, we observe that only h_v is performing worse than the others, but the value is still higher than 94% of the optimal solution.

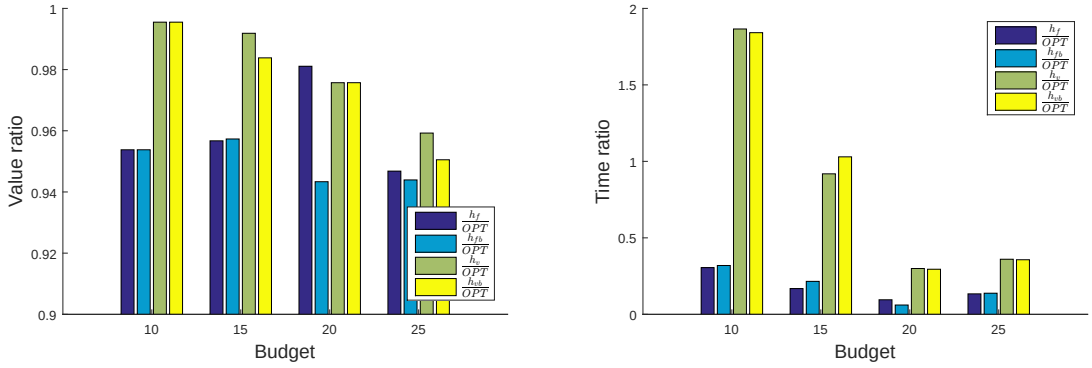
Figure 1(b) shows the time required by the heuristics w.r.t. the time needed to solve the problem exactly. As we expected, h_f, h_{fb} are faster than h_v, h_{vb}, which need to solve the problem also for each single resource. In general, the time required by PATH depends mainly on the time required to solve the exact problem with the team chosen by the heuristic. Thus, depending on the resources the team is composed of and the ways in which such resources can be allocated on the graph, the required time may vary significantly. This explains the (apparently) unusual trends of the time ratios.

[2] Even though we have already introduced and analyzed such heuristic, we report it here for the sake of completness.



(a) Utility ratios (b) Time ratios

Fig. 1. Utility and time ratios of PATH w.r.t. FORTIFY on grid graphs.



(a) Utility ratios (b) Time ratios

Fig. 2. Utility and time ratios of PATH w.r.t. FORTIFY on random graphs.

Random graphs. Looking at Figure 2, we observe that there is a clear difference both in terms of quality solution and computational time for the two main approaches of the heuristics, namely, taking into account the features of the resources or their actual values. Specifically, Figure 2(a) shows that, even though the utility ratios are all above 94%, if we consider the resources according to the utility obtained solving the exact problem, the value computed by the team they chose is higher than the corresponding ones chosen by the other two heuristics. Moreover, the budget does not seem to be a discriminating feature, being h_f, h_{fb} close to each other, and the same holds for h_v, h_{vb}. We notice that h_v returns a value equal to 96% even with a budget of 25.

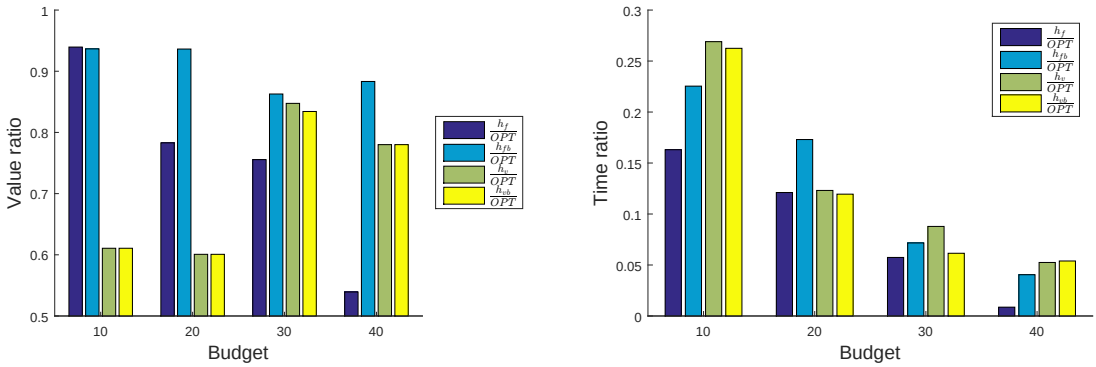
We focus now on Figure 2(b): also in this case, we notice that the budget is not a distinctive feature. Indeed, the times required by h_f, h_{fb} are similar and the same holds for h_v, h_{vb}. We observe that for low budgets the time required by the heuristics that adopts the exact values are very high. This is due to the *fixed cost* such heuristics have to pay, namely solving the problem exactly for each resource and then for the chosen team. As the budget increases, FORTIFY requires more and more time because the number of teams to be exactly evaluated grows quickly: our heuristics do not have this issue and consequently, as the budget increases, their time decreases w.r.t. the time required by FORTIFY.

5.3 Real-world comparison: protecting the Madagascar forests

We present the following model, which was built working closely with domain experts from NGOs.

Graph. We used the road and river networks used by the patrolling officers, as well as the known routes taken by groups of illegal loggers, 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 pool. 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 $b = \{5, 5, 8, 8\}$.



(a) Utility ratios (b) Time ratios

Fig. 3. Utility and time ratios of Algorithm 1 w.r.t. FORTIFY on the Madagascar graph.

Results. Differently from the synthetic instances, the results shown in Figure 3 are apparently contradictory. Indeed, Figure 3(a) shows that, for low budgets, h_v and h_{vb} are performing much worse than h_f, h_{fb}. The rationale behind this trends is that h_v and h_{vb} form teams with just one type of resource, which results being the best in terms of utility and also the cheapest. As the budget increases, such behavior is mitigated and, adding diversity to the team, the performance improve, reaching also an approximation ratio close to 85%. While the budget cannot help us in

saying whether h_v is always better than h_{vb}, it becomes very important when we take into account the features of the resources. In this case, including the budget results being the best behavior, which gives us an approximation ratio higher than 87% even with a budget of 40. If we focus on Figure 3(b), we immediately notice that all the heuristics, independently of the budget, require always less than 30% of the time employed by FORTIFY. Moreover, the higher the budget, the lower the time ratio. As for Figure 1(b), the apparently unusual trends of such ratio are due to the resolution of the problem for the chosen team.

6 Conclusions and future research

In this work, we studied the problem of preveting illegal logging in developing countries by selecting and allocating at best a team of heterogeneous resources. Due to the rising threat of this problem in South America and Africa, especially in Madagascar, and the limited budget to face this challenge, the selection, coordination and deployment of the available resources become crucial. Due to the vastness of such environments, the approach proposed in the literature is not able to face this problem when either the pool of resources that can be deployed or the budget increase. To deal with this issue, we first provide some results about the team formation problem when the function of the team is submodular. Then, we exploit such results to design a simple but very effective algorithm based on different heuristics. We test our algorithm, showing that the quality of its performance is high while being significantly fast w.r.t. the exact algorithm, allowing us to tackle bigger and closer–to–reality challenges.

In the future, we will expand our research along two dimensions. On one side, we will try to exploit the structure of the graph to improve both FORTIFY and PATH studying possible adaptations of techniques that exploit properties of the graph, e.g., min-cuts or the degree of the nodes. On the other side, we will study scenarios in which resources can be grouped together according to some of their features, e.g., the type of soil they can move on, such as ground, sea or air, making the model more realistic and exploitable for further applications.

References

1. AF&PA. www.afandpa.org/issues/issues-group/illegal-logging, 2016.
2. T. F. Allnutt, G. P. Asner, C. D. Golden, and G. V. N. Powell. Mapping recent deforestation and forest disturbance in northeastern madagascar. *TROP CONSERV SCI*, 6(1):1–15, 2013.
3. R. Andonov, V. Poirriez, and S. Rajopadhye. Unbounded knapsack problem: Dynamic programming revisited. *EUR J OPER RES*, 123(2):394–407, 2000.
4. N. Basilico, G. De Nittis, and N. Gatti. Adversarial patrolling with spatially uncertain alarm signals. *arXiv:1506.02850*, 2015.
5. N. Basilico, G. De Nittis, and N. Gatti. Multi-resource defensive strategies for patrolling games with alarm systems. *arXiv:1606.02221*, 2016.
6. N. Basilico, G. De Nittis, and N. Gatti. A security game combining patrolling and alarm-triggered responses under spatial and detection uncertainties. In *AAAI*, 2016.
7. V. Conitzer and T. Sandholm. Computing the optimal strategy to commit to. In *EC*, pages 82–90, 2006.
8. N. Dhital, R. R. Vololomboahangy, and D. P. Khasa. Issues and challenges of forest governance in madagascar. *CAN J DEV STUD*, 36(1):38–56, 2015.
9. D. Eppstein and M. T. Goodrich. Studying (non-planar) road networks through an algorithmic lens. In *SIGSPATIAL*, page 16, 2008.
10. F. Fang, P. Stone, and M. Tambe. When security games go green: Designing defender strategies to prevent poaching and illegal fishing. In *IJCAI*, 2015.
11. U. Feige. A threshold of ln n for approximating set cover. *JACM*, 45(4):634–652, 1998.
12. M. Jain, B. An, and M. Tambe. An overview of recent application trends at the AAMAS conference: Security, sustainability, and safety. *AI MAG*, 33(3):14–28, 2012.
13. M. Jain, V. Conitzer, and M. Tambe. Security scheduling for real-world networks. In *AAMAS*, pages 215–222, 2013.
14. M. Jain, D. Korzhyk, O. Vaněk, V. Conitzer, M. Pěchouček, and M. Tambe. A double oracle algorithm for zero-sum security games on graphs. In *AAMAS*, pages 327–334, 2011.
15. M. P. Johnson, F. Fang, and M. Tambe. Patrol strategies to maximize pristine forest area. In *AAAI*, 2012.
16. S. Khuller, A. Moss, and J. S. Naor. The budgeted maximum coverage problem. *INFORM PROCESS LETT*, 70(1):39–45, 1999.
17. D. Korzhyk, V. Conitzer, and R. Parr. Complexity of computing optimal stackelberg strategies in security resource allocation games. In *AAAI*, 2010.
18. S. Mc Carthy, M. Tambe, C. Kiekintveld, M. L. Gore, and A. Killion. Preventing illegal logging: Simultaneous optimization of resource teams and tactics for security. In *AAAI*, 2016.
19. C. Nellemann et al. *Green carbon, black trade: illegal logging, tax fraud and laundering in the world's tropical forests*. United Nations Environment Programme, GRID-Arendal, 2012.
20. T. H. Nguyen, F. M. Delle Fave, D. Kar, A. S. Lakshminarayanan, A. Yadav, M. Tambe, N. Agmon, A. J. Plumptre, M. Driciru, F. Wanyama, et al. Making the most of our regrets: Regret-based solutions to handle payoff uncertainty and elicitation in green security games. In *GameSec*, pages 170–191, 2015.
21. S. Okamoto, N. Hazon, and K. Sycara. Solving non-zero sum multiagent network flow security games with attack costs. In *AAMAS*, pages 879–888, 2012.
22. K. Papadaki, S. Alpern, T. Lidbetter, and A. Morton. Patrolling a border. *OPER RES*, 64(6):1256–1269, 2016.
23. L. Tacconi. *Illegal logging: law enforcement, livelihoods and the timber trade*. Earthscan, 2012.
24. M. Tambe. *Security and game theory: algorithms, deployed systems, lessons learned*. Cambridge University Press, 2011.
25. P. Toyne, C. OBrien, and R. Nelson. The timber footprint of the g8 and china: Making the case for green procurement by government. *WWF International, Version*, 2, 2002.
26. B. Von Stengel and S. Zamir. Leadership with commitment to mixed strategies. 2004.
27. WWF. www.worldwildlife.org/initiatives/stopping-illegal-logging, 2016.