A Graph-Theoretic Approach to Protect Static and Moving Targets from Adversaries

J.P. Dickerson G.I. Simari V.S. Subrahmanian
Department of Computer Science and UMIACS
University of Maryland
College Park, Maryland, USA
{jdicker1,gisimari,vs}@cs.umd.edu

Sarit Kraus
Department of Computer Science
Bar-Ilan University
Ramat Gan, Israel
sarit@cs.biu.ac.il

ABSTRACT

The static asset protection problem (SAP) in a road network is that of allocating resources to protect vertices, given any possible behavior by an adversary determined to attack those assets. The dynamic asset protection (DAP) problem is a version of SAP where the asset is following a fixed and widely known route (*e.g.*, a parade route) and needs to be protected. We formalize what it means for a given allocation of resources to be "optimal" for protecting a desired set of assets, and show that randomly allocating resources to a single edge cut in the road network solves this problem. Unlike SAP, we show that DAP is not only an NP-complete problem, but that approximating DAP is also NP-hard. We provide the GreedyDAP heuristic algorithm to solve DAP and show experimentally that it works well in practice, using road network data for real cities.

Categories and Subject Descriptors

I.2.11 [**Computing Methodologies**]: Distributed Artificial Intelligence—*Intelligent agents*

General Terms

Security, Algorithms

Keywords

Agent Systems, Game Theory, Adversarial Reasoning

1. INTRODUCTION

In this paper, we consider two problems related to the protection of assets in a road network. The first problem assumes that certain arbitrary vertices (denoting assets) in a graph (representing the road network) must be protected from adversaries who may be located at any subset of vertices. We call this the static asset protection problem (SAP) because the asset being protected is static. For example, the police in a US city may be protecting a hotel where a famous politician is staying for a few days. In contrast, the dynamic asset protection problem (DAP, for short) considers the case where the asset being protected is moving along a pre-determined route. For instance, a politician may be traveling along a parade route and the police need to protect the entire route. In both cases, the police have limited resources to protect the assets in question.

Both problems are intimately related to *network interdiction* [9, 17], where an enemy attempts to traverse a graph from a start vertex to an end vertex while an interdictor impedes his progress by "breaking" edges in the graph. Work in network interdiction has traditionally focused on stopping enemy movement along some path; however, our work is motivated by a need to *protect* a static asset's position or dynamic asset's path.

In Section 2, we formalize the static asset protection problem and define an "optimal" deployment of resources to protect the asset in question, taking adversarial behavior into account. Then, in Section 3, we develop a formal theoretical model based on minimal edge cuts in graphs and show that randomization over what we call *single* minimal edge cuts yields the optimal asset protection. We propose an algorithm for SAP and analyze its running time. Section 4 defines the dynamic asset protection problem and shows that this problem is NP-complete. We propose a greedy algorithm that tries to quickly compute a (sub-optimal) way of solving DAP. In Section 5, we describe the results of experiments we conducted using road networks drawn from real cities. These results show that our algorithms perform very well on real world data for both SAP and DAP. Related work is discussed in Section 6.

2. PRELIMINARIES

Let $G = (V, E)$ be a graph modeling a road network, where the set V of vertices represents locations, and the set E of edges represents connections between these points. We assume the existence of two sets of distinguished vertices $S, T \subseteq V$ such that $S \cap T = \emptyset$; S is the set of *source* vertices, *i.e.*, vertices from which the adversary may start, and T is the set of *target* vertices, *i.e.*, those vertices in which the adversary is interested. The assumption that we know S leads to no loss of generality because if we do not, we can simply set S to $V \setminus T$. Let *Res* be a set of *resources* to protect a target vertex — without loss of generality, we assume that any resource in *Res* can be placed along any edge and is always successful. For instance, each member of *Res* may be a police unit, suitably equipped, and the deployment of such a unit along an edge is assumed to be good enough to

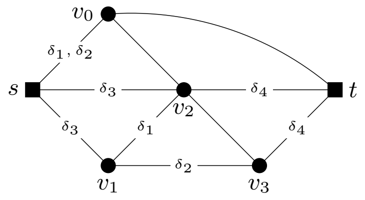
Cite as: A Graph-Theoretic Approach to Protect Static and Moving Targets from Adversaries, John P. Dickerson, Gerardo I. Simari, V.S. Subrahmanian, and Sarit Kraus, *Proc. of 9th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2010)*, van der Hoek, Kaminka, Lespérance, Luck and Sen (eds.), May, 10–14, 2010, Toronto, Canada, pp. XXX-XXX.

foil an attack.

Definition 1. *Given graph $G = (V, E)$ and set Res of resources, a* deployment *is any injective total function $\delta : Res \to E$. If SD is the set of all possible deployments, a* probabilistic deployment *is any probability distribution over SD.*

By requiring deployments to be total and injective, we are implicitly assuming that $|Res| \leq |E|$. When the graph and set of resources is understood from context, we use SPD to denote the set of all possible probabilistic deployments.

Example 1. *We now show a probabilistic deployment π over a simple road network. We consider four possible deployments $\{\delta_1, \delta_2, \delta_3, \delta_4\}$ over a graph $G = (V, E)$, with all other $\delta \in SD$ having probability 0. Each of these deployments has a certain probability of being used, forming our probabilistic deployment π. As this is a simple example, both the source set $S = \{s\}$ and target set $T = \{t\}$ have cardinality 1. $|Res| = 2$, meaning we have two resources.*



$\pi \in SPD$		
Dep.	*Resources*	*P.*
δ_1	$(s, v_0), (v_1, v_2)$	0.15
δ_2	$(s, v_0), (v_1, v_3)$	0.30
δ_3	$(s, v_1), (s, v_2)$	0.45
δ_4	$(v_2, t), (v_3, t)$	0.10

The edges are given labels representing which deployments place a resource on that edge. No label implies no $\delta_i \in \{\delta_1, \delta_2, \delta_3, \delta_4\}$ places a resource on that edge.

Definition 2. *Given a graph $G = (V, E)$, sets $S \subseteq V$ and $T \subseteq V$ such that $S \cap T = \emptyset$, and a set Res of resources, an* adversarial behavior function *is any function $\beta : SPD \times T \to paths(S, T)$, where $paths(S, T)$ is the set of potential paths from $s \in S$ to $t \in T$ in G. We refer to the evaluation of β w.r.t. $t \in T$ and a fixed $\pi \in SPD$ as $\beta_\pi(t)$.*

A *path* is a sequence of vertices $\langle v_0, v_1, \ldots, v_k \rangle$ such that for all $0 \leq i < k$, $(v_i, v_{i+1}) \in E$. From Definition 2, the probability that the adversary's attack on edge e is stopped by a probabilistic deployment π is:

$$P\left(\begin{array}{c} \textit{Adversary stopped} \\ \textit{at edge } e \textit{ given } \pi \end{array}\right) = \sum_{\delta \in SD \ s.t.\ \exists r_i \in Res, \delta(r_i) = e} \pi(\delta)$$

where $\pi(\delta)$ is the probability that we will use deployment δ to protect our asset. The above expression looks at all deployments δ that place a resource along edge e; each of these deployments will stop an attack involving edge e. The sum of the probabilities of all such deployments is the probability that a given probabilistic deployment π will stop an attack. We now define what it means for a deployment to fail against an attack.

Definition 3. *A deployment $\delta \in SD$* fails against *a path $p \in paths(S, T)$ iff $\forall e \in p$ there exists no $r \in Res$ s.t. $\delta(r) = e$. We say p is* stopped *by δ if δ does not fail against p.*

Intuitively, this means that none of the resources are deployed at any of the edges that are part of the path chosen by the adversary, given our probabilistic deployment π. From this, we see the probability that an attack by the adversary on any $t \in T$ will *not* be stopped given a probabilistic deployment π and an adversarial behavior function β is:

$$P(\beta_\pi(t) \textit{ not stopped}) = \sum_{\delta \in SD \ s.t.\ \delta \textit{ fails against } \beta_\pi(t)} \pi(\delta) \quad (1)$$

The adversary's goal is to compute an adversarial behavior function that best responds to our probabilistic deployment. Our goal is to compute a probabilistic deployment that is most effective against the adversary.

Problem 1 (Adversary): Given probabilistic deployment π, find an adversarial behavior function β^* that maximizes

$$\max_{t \in T} \left(P(\beta^*_\pi(t) \textit{ not stopped})\right) \quad (2)$$

Problem 2 (Reasoning Agent; Static Asset Protection Problem): Given any adversarial behavior function β, find a probabilistic deployment π^* that minimizes

$$\max_{t \in T} \left(P(\beta_{\pi^*}(t) \textit{ not stopped})\right) \quad (3)$$

π^* is called an *optimal SAP-deployment.*

3. COMPUTING OPTIMAL PROBABILISTIC DEPLOYMENTS FOR SAP

We now focus on computing an optimal SAP-deployment. While this problem has been investigated by Wood [17, 16], our probabilistic deployment-based approach provides a stepping stone to the more complex dynamic problem tackled later in the paper. We start with a simple observation which says that we only need to consider acyclic attack paths.

Proposition 1. *For every cyclic path $p \in paths(S, T)$, there exists an acyclic path $p' \in paths(S, T)$ such that*

$$P(p' \textit{ stopped by } \pi) \leq P(p \textit{ stopped by } \pi)$$

where π is any probabilistic deployment.

From Proposition 1, we see that the set of possible attack paths is finite, meaning it is possible to compute probabilities for all paths.

Using Minimal Edge Cuts. We now discuss methods for computing probabilistic deployments that are based on obtaining *edge cuts* of a graph.

Definition 4. *Let $G = (V, E)$ be a graph, and $s, t \in V$. We say that $C \subseteq E$ is a* local edge cut *of G w.r.t. s and t if and only if there is no path from s to t in $G' = (V, E \setminus C)$. Furthermore, if there is no local edge cut C' of G such that $|C'| < |C|$, we say that C is a* minimal local edge cut *of G w.r.t. s and t.*

This is a standard definition from graph theory [8] which is sometimes also called *minimum s-t edge cut.* We deviate slightly from the usual notion of "minimal", since we use *cardinality* instead of set inclusion. Informally, a probabilistic deployment is edge cut-based iff there is a minimal edge cut such that every deployment with a non-zero probability assigns resources to edges in that edge cut.

Definition 5. *Let $G = (V, E)$ be a graph, $s, t \in V$, and Res be a set of resources. We say that a probabilistic deployment π is* edge cut-based w.r.t. s and t *if and only if any deployment δ for which $\pi(\delta) \neq 0$ is such that $\{e \mid \delta(e) = r$ for some $e \in E$ and $r \in Res\} \subseteq C$, for some minimal local edge cut C w.r.t. s and t in G.*

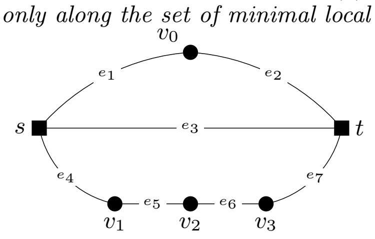
The following result shows that to find a probabilistic deployment that best protects the adversary's targets, only edges that belong to minimal edge cuts w.r.t. nodes $s \in S$ and $t \in T$ need to be protected.

Theorem 1. *Let $G = (V, E)$ be a graph, $s, t \in V$, Res be a set of resources, and β be an adversarial behavior function. If the size of a minimal local edge cut w.r.t. s and t is greater than or equal to $|Res|$, then for any non-edge cut-based probabilistic deployment π there exists an edge cut-based probabilistic deployment π_{cut} such that*

$$P(\beta \textit{ not stopped by } \pi_{cut}) \leq P(\beta \textit{ not stopped by } \pi).$$

Proof. Let π be a non-edge cut-based probabilistic deployment. Therefore, there exists deployment δ s.t. $\pi(\delta) > 0$, and $Z = \{e \mid \delta(e) = r \textit{ for some } e \in E \textit{ and } r \in Res\} \not\subseteq C$, for all minimal local edge cuts C w.r.t. s and t in G. Let C be some minimal local edge cut and let edge $e \in Z$ be such that $e \notin C$, and consider the path $\beta(\pi)$; clearly, since C is a minimal edge cut w.r.t. s and t, $C \cap \beta(\pi) \neq \emptyset$. However, since $|C| > |Res|$, it must be the case that there exists some edge $e' \in C$ such that $e' \notin Z$, and therefore there exists a probabilistic deployment π' that is identical to π except that $\pi'(\delta) = 0$ and $\pi'(\delta') = \pi(\delta)$, where $\delta'(e') = \delta(e)$ and $\delta'(e)$ is null. Note that δ' is protecting edge e' whereas δ was not and, since π' assigns the probability mass assigned to δ by π to δ' instead, and δ fails against $\beta(\pi)$ whenever δ' does (but not necessarily vice versa), it must be the case that $P(\beta \textit{ not stopped by } \pi') \leq P(\beta \textit{ not stopped by } \pi)$. If we continue doing this until there are no more edges in Z that do not belong to C, we will have obtained an edge cut-based probabilistic deployment π_{cut} s.t. $P(\beta \textit{ not stopped by } \pi_{cut}) \leq P(\beta \textit{ not stopped by } \pi)$. □

Example 2. *We are interested in finding a probability distribution π that minimizes Equation 3. As Theorem 1 states, all deployments δ s.t. $\pi(\delta) > 0$ will deploy resources only along the set of minimal local edge cuts.*



Set of Min s-t Cut Sets	
(e_1, e_3, e_4)	(e_2, e_3, e_4)
(e_1, e_3, e_5)	(e_2, e_3, e_5)
(e_1, e_3, e_6)	(e_2, e_3, e_6)
(e_1, e_3, e_7)	(e_2, e_3, e_7)

Calculating the set of minimal $s - t$ edge cuts of this example is easy. There exist three clear edge-disjoint paths from source node s to target node t: $\langle s, v_0, t\rangle$, $\langle s, t\rangle$, $\langle s, v_1, v_2, v_3, t\rangle$. The set of minimal edge cuts consists of all combinations of one edge from each of the three paths. Using this tabulation of edge cuts, we can form a SAP-optimal distribution over deployments that place resources on edges in cuts in our set of local minimal edge cuts.

Theorem 1 says that deploying resources over minimal edge cuts guarantees that the resulting probabilistic deployment will be SAP-optimal. However, for graphs that have more than one minimal edge cut, it says nothing about which min-cut to use. It turns out that deploying resources over edges belonging to *any single minimal edge cut* is sufficient for optimality. Therefore, similar to Definition 5, we have the concept of a *single* edge cut-based probabilistic deployment, which is based on a single minimal edge cut.

Definition 6. *Let $G = (V, E)$ be a graph, $s, t \in V$, and Res be a set of resources. We say that a probabilistic deployment π is* single-edge cut-based *w.r.t. s and t if and only if there exists a minimal local edge cut C w.r.t. s and t in G such that any deployment δ for which $\pi(\delta) \neq 0$ is such that $\{e \mid \delta(e) = r$ for some $e \in E$ and $r \in Res\} \subseteq C$.*

We can now prove the following result.

Theorem 2. *Let $G = (V, E)$ be a graph, $s, t \in V$, Res be a set of resources, and β be an adversarial behavior function. If the size of a minimal local edge cut w.r.t. s and t is greater than or equal to $|Res|$, then for any edge cut-based probabilistic deployment π_{cut} there exists a single-edge cut-based probabilistic deployment π_{single} such that*

$$P(\beta \textit{ not stopped by } \pi_{single}) = P(\beta \textit{ not stopped by } \pi_{cut}).$$

Proof. Let π_{cut} be a non-single-edge cut-based probabilistic deployment, and C be a minimal edge cut w.r.t. s and t in G; without loss of generality, we can assume that all deployments δ for which $\pi_{cut}(\delta) > 0$ are such that $\{e \mid \delta(e) = r \textit{ for some } e \in E \textit{ and } r \in Res\} \subseteq C$ (let Δ_C be the set of all such deployments), *except* for one distinguished $\delta' \notin \Delta_C$, which is s.t. $\{e \mid \delta'(e) = r \textit{ for some } e \in E \textit{ and } r \in Res\} \subseteq C'$, for some minimal edge cut C' w.r.t. s and t in G, where $C \neq C'$. Now, let π_{single} be identical to π_{cut}, except that $\pi_{single}(\delta') = 0$, and $\pi_{single}(\delta^*) = \pi_{cut}(\delta')$, for some $\delta^* \in \Delta_C$. Clearly, since C and C' are both minimal edge cuts, δ' fails against β if and only if δ^* does, and therefore we have that $P(\beta \textit{ not stopped by } \pi_{single}) = P(\beta \textit{ not stopped by } \pi_{cut})$. Since π_{single} is a single-edge cut-based probabilistic deployment, the result follows. □

An important consequence of Theorem 2 is that an algorithm for computing an optimal probabilistic deployment need not compute all possible minimal edge cuts, since deploying resources over any of them exclusively is a sufficient condition for SAP-optimality.

Equivalence of Deployment- and Edge-Based Strategies. Although the above result significantly reduces the search space, its worst-case complexity remains exponential in the number $m = |C|$ of edges in some minimal local edge cut C and $k = |Res|$, the number of resources available, because we must focus on $\binom{m}{k}$ possible deployments. Fortunately, we do not have to compute all such deployments, since we can express probabilistic deployments directly as probability distributions over edges. In the future, we refer to such distributions as *edge-based probabilistic deployments.* We now prove that *uniform* edge-based probabilistic deployments and (conventional) uniform single-edge cut-based probabilistic deployments are equivalent in terms of their effectiveness in stopping the adversary; we also show that uniform edge-based probabilistic deployments can be computed in polynomial time in the size of the cut.

Theorem 3. *Let $G = (V, E)$ be a graph, $s, t \in V$, Res be a set of resources, C be an arbitrary minimal edge cut,* *and π_{single} be an optimal uniform single-edge cut-based probabilistic deployment w.r.t. s and t over C. Then, placing resources uniformly at random over edges $e \in C$ is equivalent (*i.e.*, yields the same probability in Equation 3) to choosing a deployment based on π_{single}. Furthermore, this edge-based strategy can be computed in PTIME.*

PROOF. First of all, note that a non-uniform single-edge cut-based probabilistic deployment is sub-optimal; we will sketch the proof of this statement first. The reasoning agent's goal is to minimize the maximum probability of a successful attack against a target, over all targets (Equation 3). Although a non-uniform distribution over all single-edge cut-based deployments could protect a subset of targets better than a uniform distribution would (Equation 1), this non-uniform distribution would potentially increase the probability of successful attack against a different subset of targets. Thus, the maximal probability of successful attack over all targets would increase.

To show the equivalence, let $m = |C|$, $k = |Res|$, and $B = \binom{m}{k}$ be the number of deployments of k resources over m edges. We will show that the probability of any resource being placed on an edge $e \in C$, referred to as $P_{res}(e)$, is the same under both edge- and deployment-based strategies. Using an edge-based probabilistic deployment ξ, we randomly place k resources on m edges. Clearly, $P_{res}(e) = \frac{k}{m}$. If instead we followed the deployment-based π_{single}, we can compute:

$$P_{res}(e) = \sum_{\delta \in SD} \left(\pi_{single}(\delta) \cdot \frac{\binom{m}{k} - \binom{m-1}{k}}{\binom{m}{k}} \right) \quad (4)$$

$$= B\left(\frac{1}{B} \cdot \frac{k}{m}\right) = \frac{k}{m} \quad (5)$$

Note that in Equation 4 above we are summing over B possible deployments, and that $\pi_{single}(\delta_i) = \frac{1}{B}$ is uniformly distributed. The fractional term in this equation simply represents the deployments that assign a resource to e out of all possible deployments over edges in C. The step from Equation 4 to 5 comes from applying the identity $\binom{m}{k} = \binom{m-1}{k-1} + \binom{m-1}{k}$ and simplifying the resulting expression.

As both strategies guarantee identical coverage of all edges in C, they provide identical protection of our target set T. Furthermore, as the edge-based strategy must only randomly choose some k-subset of C, it can be computed in time in $O(|C|)$. $\square$

The above result yields a PTIME algorithm (shown in Figure 1) to compute a SAP-optimal deployment. It is well known that finding (minimal or otherwise) local edge cuts w.r.t. s and t can be accomplished by running the Edmonds-Karp [4] algorithm and selecting minimal sets (w.r.t. set inclusion) of vertices reachable from s in the final residual graph. The Edmonds-Karp algorithm has a running time in $O(|V| \cdot |E|^2)$. Other related methods, like Dinic's algorithm [7], offer better performance ($O(|V|^2 \cdot |E|)$ or better). Though these algorithms apply to directed graphs, we can apply them to the SAP problem by replacing each undirected edge with two oppositely facing directed edges [6]. The result below states that this algorithm is correct.

PROPOSITION 2. *ComputeEdgeBasedDeployment (Fig. 1) correctly computes a SAP-optimal probabilistic deployment.*

```
algorithm ComputeEdgeBasedDeployment($G = (V, E)$, $s$, $t$)
1.  $C :=$ any minimal edge cut from $s$ to $t$ in $G$;
2.  Let $\xi$ be a probability distribution over $E$;
3.  Set $\xi(e) = 0$ for all $e \in E$;
4.  Set $\xi(e) = \frac{1}{|C|}$ for all $e \in C$;
5.  Return $\xi$;
```

Figure 1: Computing edge-based deployments.

PROOF. *Sketch.* Theorem 3 proves that randomly placing resources over a minimal s-t edge cut is equivalent to randomly choosing a deployment based on π_{single}, a single-edge cut-based probabilistic deployment. Theorem 2 shows an equivalence between π_{single} and the more general edge cut-based probabilistic deployment π_{cut}. Finally, Theorem 1 shows that it is never better to choose some generic probabilistic deployment π over the cut-based π_{cut}. $\square$

Theorem 3 and Proposition 2 show that ComputeEdgeBasedDeployment (Figure 1) computes a SAP-optimal deployment in polynomial time.

4. PROTECTING A MOVING TARGET

Suppose a politician is participating in a widely advertised parade from location A to B. What is the safest parade route, *i.e.*, the route that would provide the most effective protection, given that the actual route is known to adversaries in advance? We need a mechanism to select a path between his origin and the destination, and a probabilistic deployment of resources such that the chance of an adversary successfully attacking along the path is minimized.

DEFINITION 7. *Given a graph $G = (V, E)$, set of adversarial source nodes $S \subseteq V$, and target nodes $t_s, t_e \in (V \setminus S)$, a route r is a simple path $\langle t_s, t_0, \ldots, t_k, t_e \rangle$ from t_s to t_e, with $t_i \notin S$. Furthermore, we define the set of targets w.r.t. this route as $T_r = \{t_s, t_0, \ldots, t_k, t_e\}$.*

Intuitively, this means that every node on the route from a starting position t_s to a desired ending point t_e is considered a potential target for attack. Given a route r from t_s to t_e, Figure 1 gives us a PTIME algorithm to compute an optimal probabilistic deployment π_r to protect the set of targets T_r.
Dynamic Asset Protection (DAP) Problem: Given t_s and t_e, select route r_* such that

$$r_* = \underset{r \in routes(t_s, t_e)}{\operatorname{argmin}} P\left(\beta_{\pi_r} \textit{ not stopped}\right) \quad (6)$$

where $routes(t_s, t_e)$ is the set of all routes from t_s to t_e, π_r is the optimal probabilistic solution w.r.t. T_r for a route r, and β_{π_r} is an optimal adversarial strategy against π_r calculated by maximizing Equation 2. Note that Algorithm 1 allows us to calculate π_r before calculating β_{π_r}. We now extend the SAP algorithms developed in the last section to a solution for the DAP problem using minimal edge cuts.

PROPOSITION 3. *Calculating a route r_* (called a DAP-optimal route) such that*

$$r_* = \underset{r \in routes(t_s, t_e)}{\operatorname{argmin}} |C_r|$$

where C_r is the minimal $S-T_r$ edge cut is equivalent[1] *to calculating the optimal route r_{best} using Equation 6.*

PROOF. Assume that route r_* is not DAP-optimal, with $a = |C_{r_*}|$. Then there is an optimal route r_{best} such that $b = |C_{r_{best}}|$ and $b > a$. Let $R = |Res|$, and π_{r_*} and $\pi_{r_{best}}$ be the optimal probabilistic deployments computed for each of these routes, respectively. Then the probability of an edge e in the cut C_{r_*} being covered by π_{r_*} is $\frac{R}{a}$, while the probability of an edge e' in the cut $C_{r_{best}}$ being covered by $\pi_{r_{best}}$ is $\frac{R}{b}$. Recall $b > a$, so $\frac{R}{b} < \frac{R}{a}$; that is, $\pi_{r_{best}}$ provides worse edge cut coverage than π_{r_*}. This contradicts our assumption of the optimality of route r_{best}. $\square$

Proposition 3 shows that the problem of finding a DAP-optimal route r is equivalent to finding a route r' such that the size of the minimal $S-T_{r'}$-edge cut is minimized over all routes in *routes*(t_s, t_e). We now show that the decision version of this problem is NP-complete.

PROPOSITION 4. *Let $G = (V, E)$ be a graph, $t_s, t_e \in V$, $S \subseteq V$, and $k \in \mathbb{N}$. Deciding if there exists a route $r \in$ routes(t_s, t_e) such that the size of a minimal $S-T_r$ edge cut is k is NP-complete.*

PROOF. *Membership in NP*: Clearly, given a route $r \in$ *routes*(t_s, t_e), we can verify in polynomial time if the size of a minimal edge cut is k or not by means of a max-flow algorithm such as Edmonds-Karp.
NP-hardness: We shall reduce the SUBSET-SUM (SS) problem with positive integers to our problem in polynomial time in order to prove NP-hardness. This problem involves deciding, given a set $P = \{p_1, \dots, p_n\}$ of positive integers and an integer c, if there exists $P' \subseteq P$ such that $\sum_{p_i \in P'} p_i = c$. Given an instance of SS, we must then provide an instance of our problem such that its solution provides an answer to SS for the original instance if and only if one exists.

Assume then that we have an instance of SS as described above. Construct a graph $G = (V, E)$ as shown in Figure 2; we have two nodes in V for each $p_i \in P$ (we will refer to these as *pair nodes*, and to each one in particular as *bottom* and *top*), and three more nodes t_s, t_e, and s. Furthermore, we have $\sum_{p_i \in P} p_i$ more nodes, arranged as in the figure, so that a set of p_i nodes corresponds to the pair nodes associated with p_i; we shall call this last set of nodes "gadget nodes". The total number of nodes in V is therefore $2n + 3 + \sum_{p_i \in P} p_i$. Next, we add edges from t_s to the first pair nodes, from the last pair nodes to t_e, and from each remaining pair node to the two pair nodes on its right. Finally, we add an edge from s to each gadget node, and from each gadget node to the bottom node of its corresponding pair. This yields a total of $4n + 2 \cdot \sum_{p_i \in P} p_i$ edges. We will now show that a solution exists for the instance of SS if and only if G has a route r such that the size of a minimal $s - T_r$ edge cut is $c + 3n - 2$.
($\Rightarrow$) Suppose there exists a solution $P' = \{p'_1, \dots, p'_k\}$ to the SS instance, which means that $\sum_{p'_i \in P'} p'_i = c$. Consider then the route $r = \{t_s, \dots, u_i, \dots, t_e\}$, where u_i corresponds to the bottom node in each pair if $p_i \in P'$ and to the top node if $p_i \notin P'$. Let $\mathcal{C}_r$ be a minimal $s - T_r$ edge cut. Clearly, removing the minimal number of edges from G such that s and T_r are disconnected can be accomplished by removing, for each node $u_i \notin \{t_s, t_e\}$, the four

[1]*i.e.*, r_*'s probability of stopping β_{π_r} is the same whether computed using Equation 5 or using the above formula

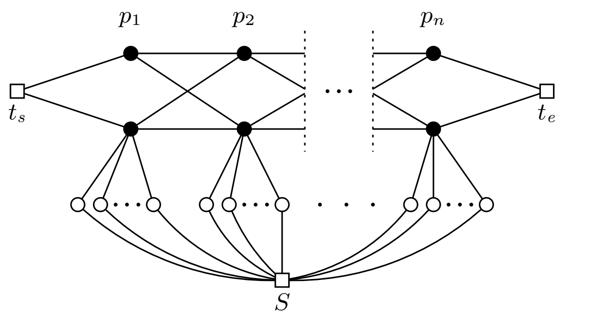


Figure 2: Graph used in the proof of Proposition 4

edges that connect them to the previous and the following pair nodes (these are actually three for the first and last nodes) and, in the case of bottom nodes in the route, all p'_i edges connecting to gadget nodes as well. Intuitively, all nodes add four edges to the cut, except the first and last which add three, and each bottom node in the route adds p'_i more edges. However, this reasoning is double counting edges; looking closely, we see that each pair node actually adds three edges (the one not added by the previous pair, plus the two connecting to the pair on the right), except for the last one, which adds only one. This brings the total size of the cut to $\left(\sum_{p'_i \in P'} p'_i\right) + 3(n-1) + 1 = c + 3n - 2$.
($\Leftarrow$) Suppose there exists a route r with the above characteristics; by hypothesis, the minimal $s - T_r$ edge cut is of size $c + 3n - 2$. If we take the set of nodes in r that are bottom nodes in a pair, we can see that the sum of incoming edges from gadget nodes is exactly c. Therefore, the corresponding solution to SS consists of the integers p'_i corresponding to such pairs in the construction of G.
Since the reduction can be performed in polynomial time, we have proved that the problem is NP-hard. $\square$

As this result shows that computing DAP-optimal routes is intractable,[2] we wondered if there are PTIME approximation algorithms for DAP; unfortunately, the answer is "no". The proof first requires a lemma.

LEMMA 1. *Let $G = (V, E)$ be a graph, $t_s, t_e \in V$, $S \subseteq V$, and $r \in$ routes(t_s, t_e) be an optimal route. Then, we have:*

1. *There exists a $\{t_s, t_e\}$–S edge cut C^* such that $|C^*| = |C|$, and t_s, t_e belong to the same connected component in $G' = (V, E \setminus C^*)$.*

2. *Let $C_{s\text{-}e}$ be a minimal $\{t_s, t_e\}$–S edge cut; then, $|C_{s\text{-}e}| \leq |C| \leq \sum_{s \in S}$ degree(s).*

PROOF. The proof of the first part follows directly from the definition of minimal edge cut and the problem statement. For the second part, consider the first inequality. Clearly, since C is a minimal T_r–S edge cut and $\{t_s, t_e\} \subseteq T_r$, the inequality holds. For the second one, it is sufficient to note that no cut can be larger than the set of edges that isolates all $s \in S$. $\square$

[2]Note that this result can easily be extended to planar graphs by replacing each pair of crossing edges in Figure 2 with a node at the crossing point and four corresponding edges. The proof can then be modified to contemplate this new graph, meaning that even for the more restricted class of planar graphs the problem remains NP-complete.

The above Lemma shows that DAP is closely related to the problem of dividing the set of vertices of a graph into two disconnected sets of a certain size while removing the minimum possible number of edges. This problem is NP-complete [11] — furthermore, Bui and Jones [3] show that approximating either edge or vertex separations on a general graph is intractable. Specifically, given an optimal edge separator C, finding an approximate solution C' such that $|C'| \leq |C| + |V|^{2-\epsilon}$, with $\epsilon > 0$, is NP-hard. As DAP is closely related to this problem, we can prove that any good approximation algorithm will also be intractable. In the following, we refer to the problem of finding a partition P of size k such that t_e is reachable from t_s in P by means of a path $p_k \in P$ and such that the minimal edge cut associated with P is the smallest among all such partitions as the "k-DAP problem".

THEOREM 4. *Let $G = (V, E)$ be a graph, $t_s, t_e \in V$, $S \subseteq V$, and $k \in \mathbb{N}$. Given an optimal edge cut C for the k-DAP problem, finding an approximate solution C' such that $|C'| \leq |C| + |V|^{2-\epsilon}$, with $\epsilon > 0$, is NP-hard.*

PROOF. Suppose towards a contradiction that there exists an approximation algorithm that obtains a good approximation (as stated in the theorem) for the k-DAP problem. In this case, let $G' = (V, E \cup (t_s, t_e))$ be the graph obtained from G by adding an edge from t_s to t_e (if this edge was already there, then $G = G'$). Now, an edge cut C^* satisfies the conditions in the theorem for the k-DAP problem over G' if and only if C^* satisfies those same conditions for the constrained optimal k-partition problem over G (where the constraints state that t_s and t_e must belong to the same partition) since the existence of the path is guaranteed by the additional edge. This is a contradiction, since it was proved [3] that such an approximation cannot exist. The contradiction stemmed from assuming the existence of an approximation algorithm that obtains solutions satisfying the conditions, and thus no such algorithm exists. □

```
algorithm GreedyDAP($G = (V, E)$, $t_s$, $t_e$, $S$)
1. Starting at $t_e$, perform BFS to calculate
              shortest distance from all $v \in V$ to $t_e$;
2. Define heuristic functions $g, h : V \rightarrow \mathbb{N}$;
3. For each vertex $v$ in $V$ do
4.      $g(v) := degree(v)$;
5.      $h(v) := d(v, t_e)$, shortest distance from $v$ to $t_e$;
6. Let $r := A^*(t_s, t_e)$, the route returned
              by $A^*$ using functions $g$, $h$;
7. Let $C$ be any minimal edge cut from $S$ to $T_r$ in $G$;
8. Return $\{r, C\}$;
```

Figure 3: The GreedyDAP Algorithm

Greedy DAP Algorithm. Figure 3 presents a greedy DAP algorithm based on intuitions from Lemma 1. The heart of the algorithm is the execution of the A* algorithm [10] using specific cost and distance functions. For a vertex v, the path-cost function is based on *degree*(v), while the admissible heuristic estimate of the distance from v to ending target t_e is computed via a one-time breadth-first search (BFS). Computing this path-cost function is in PTIME; furthermore, as A* expands each vertex at most once, the entire search runs in PTIME. A* returns a suggested t_s, t_e-route r, so the minimal $S - T_r$-edge cut can be computed using the Edmonds-Karp min cut algorithm. As the PTIME operations of A*, BFS, and Edmonds-Karp are executed once, the greedy algorithm shown in Figure 3 is also in PTIME.

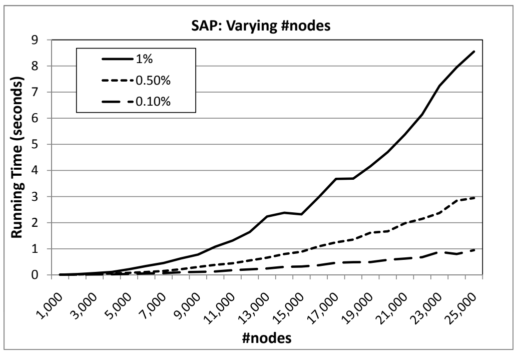


Figure 4: Running times of the algorithm for the static problem for different #sources and #targets as percentages of $|V|$.

5. EXPERIMENTAL RESULTS

We conducted experiments using a prototype JAVA implementation consisting of roughly 2,100 lines of code, relying on the JGraphT[3] library for operations on graphs. All experiments were run on multiple multi-core Intel Xeon E5345 processors at 2.33GHz, 8GB of memory, running the Scientific Linux distribution of the GNU/Linux operating system, kernel version 2.6.9-55.0.2.ELsmp. *We note that this implementation makes use of only one processor and one core.* All runs were performed over graphs corresponding to real road networks [12]. In order to obtain graphs of a given size, a seed node was randomly chosen, and neighbors progressively added until the desired number of nodes was reached. The average degree of nodes in the graphs used was approximately 2.45, which means that the number of edges was less than twice the number of nodes. Finally, all graphs plot average values over 20 to 100 runs, in order to minimize experimental error.

SAP Running Time. The first experiment focuses on comparing the running time (*cf.* Figure 4) of our SAP algorithm as we vary the number of vertices in the road network and we vary the percentage of source/target vertices in the graph. Three curves are plotted, each corresponding to different amounts of randomly selected source and target nodes as percentages of the total set of vertices (0.1%, 0.5%, and 1%). For instance, the full line corresponds to the case in which 1% of the nodes were randomly chosen as sources, another 1% were chosen as targets, and the remaining 98% were regular vertices. We can see that the number of source and target nodes greatly affects the running time of the algorithm, which is a natural consequence of the fact that the larger these sets are, the more augmenting paths will be involved in the necessary maximum flow computations.

DAP Problem: Efficiency of GreedyDAP. Figure 5 shows a comparison of the running times of a brute force algorithm and GreedyDAP. Clearly, the brute force solution does not scale well as it took more than three minutes for

[3]http://jgrapht.sourceforge.net/

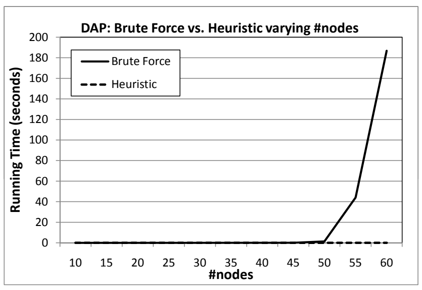


Figure 5: Comparison of running times of the brute force and the heuristic algorithms as #nodes is varied.

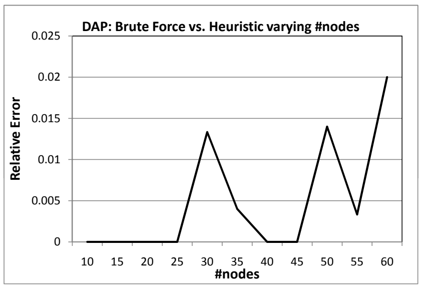


Figure 6: A plot of the relative error in the size of the cuts found by the heuristic algorithm compared to those found by brute force.

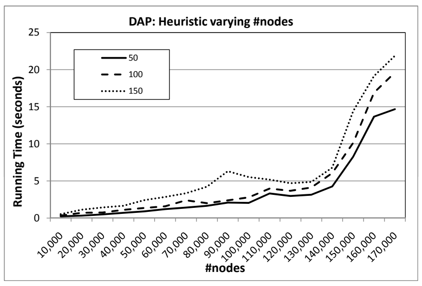


Figure 7: Running time of the heuristic algorithm as #nodes takes larger values than in Figure 5.

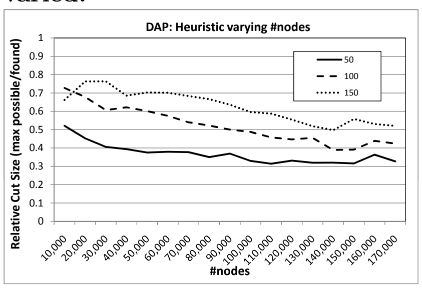


Figure 8: A plot of the cut sizes found by the heuristic algorithm relative to the upper bound cut size (lower is better).

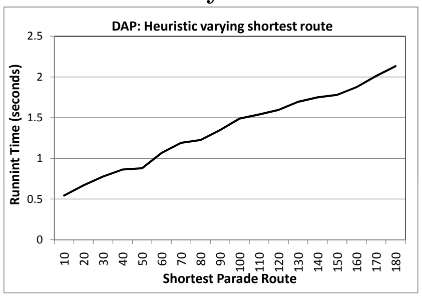


Figure 9: Running time of the heuristic algorithm as the length of the shortest parade route is varied.

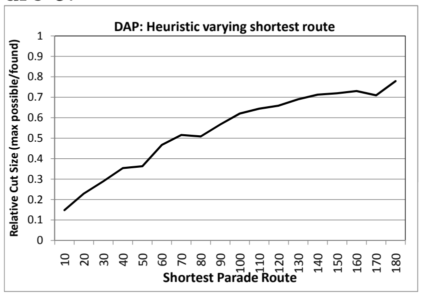


Figure 10: Relative cut sizes found by the heuristic algorithm (as in Figure 8), as length of shortest parade route varies.

a graph of 60 nodes, with 2 sources and 2 targets. These runs allowed us to compare the accuracy of GreedyDAP computed as relative error w.r.t. the size of the cuts found (absolute value of the difference divided by size of the cut found by brute force). For these runs, the error associated with GreedyDAP was at most 0.02, as shown in Figure 6.

DAP Problem: Scalability of GreedyDAP. Figure 7 analyzes the scalability of the GreedyDAP algorithm for larger graphs; in this and the remaining experiments, the number of sources was kept fixed at 20. We plotted three curves simultaneously in this figure, which differ only in the length of the minimum route (50, 100, and 150). As expected, the running time is affected by the length of the shortest route as the search space for the heuristic algorithm is made larger. Furthermore, we see that the running times all have an inflection point at 140K nodes, but still remain between 15 and 23 seconds even for 170K nodes. This can be explained by recalling that the heuristic algorithm computes the shortest route at each iteration, an operation that for the graphs used here takes time in $(|V|)$. Figure 8 also reports runs in which the number of nodes was varied, but plots the relative cut sizes found by GreedyDAP instead (also for three different shortest parade route lengths). Relative cut sizes were computed as the size of the cut found by the algorithm divided by the upper bound on such size obtained by computing a minimal $S - \{V \setminus S\}$ edge cut (*i.e.*, isolating all the source nodes from the rest of the graph). We again see that performance worsens as the length of the shortest parade route becomes longer; this is due to the fact that the number of targets (intermediate nodes) that must be protected becomes larger. Surprisingly, performance becomes better as the number of nodes becomes larger; an explanation of this behavior is that the number of sources was kept constant and thus the parade represents a smaller portion of the graph.

DAP Problem: Performance of GreedyDAP with Increasing Route Length. Finally, the two remaining figures correspond to experiments in which the number of nodes was fixed at 40K, and the length of the *shortest route* was varied. Figure 9 plots the time taken by the GreedyDAP algorithm to compute a solution as this length increases; in accordance with the results showed in Figure 4 and 7, it is more costly to compute a solution the longer the minimum route is since more targets must be protected; however, even for routes of length 180, the time taken to find a route was just over 2 seconds. In Figure 10, we plot relative cut size as a function of length of the shortest route. Quite interesting is the fact that the quality of the results returned by the heuristic algorithm decays quite gradually, and is fairly good even for relatively long route lengths such as 100, where it found on average a cut with a relative size w.r.t. the upper bound of little over 0.6. This decay is explained by the fact that, as route length increases, the algorithm has more chances of making sub-optimal choices.

6. RELATED WORK

In the past, there has been considerable interest in both interdicting a network [5, 16, 17] and protecting a set of targets [1, 2, 14, 15, 13]. While all of these papers discuss problems that can be compared to SAP, they are different in three major respects: none of them use deployment-based strategies like we do, all of them use the notion of "turns" for different players (this is not part of the problems we address), and *none of them address the DAP problem.* We now consider these papers in turn.

In deterministic network interdiction, an enemy attempts to maximize flow across a directed network while an interdictor minimizes this maximum flow by blocking edges. Wood shows that a generalized version of SAP (as a network interdiction problem with weighted edges and resources) is NP-complete [17]. In [16], a problem equivalent to SAP is given in matrix game form and shown to be solvable in polynomial time. [5] addresses a stochastic interdiction problem similar to SAP where successful edge interdiction is modeled as a random variable, but does not give a PTIME solution.

[1] and [2] consider the problem of detecting penetrations across the path(s) of patrolling robots. These robots apply a non-deterministic patrol scheme such that their movement is characterized by a probability p. [1] presents an optimal polynomial time algorithm for finding this p such that the minimal probability of penetration detection across the path is maximized, assuming the agent has full knowledge of the environment and the robot's strategy. [2] extends this model by considering adversaries with different knowledge.

[13] assumes knowledge of the adversary's behavior is accommodated by manipulating a probability distribution over states, indicating probability of that state being targeted. They discuss randomizing an MDP (or POMDP) policy in order to avoid being modeled by an opponent, while maintaining an expected reward above a certain threshold. Even though this approach could be used to solve SAP or DAP, expressing these problems in terms of MDPs would be highly inefficient because of the exponential number of states used. No graph structures are considered and randomization is over the space of MDP policies, not edge cuts.

[14] proposes models based on bounded rationality, which assumes that the follower agents will choose a strategy that is within some epsilon of the actual best, and observability limitations, in which the follower is assumed to start out with no knowledge of the desired distribution and thus begins with a uniform distribution (inspired by human behavior), which is thereafter updated slowly given observations (this is called Anchoring Theory). The authors solve Stackelberg games (in general) under the assumption of suboptimal adversaries. As in the previous case, SAP or DAP could in theory be solved by this approach, but the number of possible actions would make this game-theoretic approach intractable. Finally, [15] is also based on Stackelberg games, where the opponent is assumed to be rational. The main problem is to solve the game for the defender, for the special case of "transportation networks" (basically protecting commercial flights). The same discussion as for the previous paper applies w.r.t. SAP and DAP.

7. CONCLUSIONS

In this paper, we study the *static asset protection* (SAP) problem first and show that optimally protecting static assets with a given set of resources can be best done by first selecting a minimal cut of the graph and then randomizing where our protective resources are placed along edges in the min-cut. We show that this algorithm performs very well in practice on real world road network data.

We then introduce the *dynamic asset protection* (DAP) problem to protect moving assets whose routes are known in advance. Given a set of resources, what parade route should be picked (to afford the best degree of protection) and how best should we protect this route? We show that the DAP problem is not only NP-complete, but approximating it is also NP-hard. As a consequence, we develop a heuristic algorithm called GreedyDAP and show experimentally that it performs well in practice on real world road data.

Acknowledgments. This research is supported by NSF Grant #0705587 and ISF Grant #1685.

8. REFERENCES

[1] N. Agmon, S. Kraus, and G. Kaminka. Multi-robot perimeter patrol in adversarial settings. In *Proc. of ICRA*, pages 2339–2345, 2008.

[2] N. Agmon, S. Kraus, G. Kaminka, and V. Sadov. Adversarial Uncertainty in Multi-Robot Patrol. In *Proc. of IJCAI*, pages 1811–1817, 2009.

[3] T. Bui and C. Jones. Finding good approximate vertex and edge partitions is NP-hard. *Information Processing Letters*, 42(3):153–159, 1992.

[4] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms, Second Edition.* The MIT Press, September 2001.

[5] K. Cormican, D. Morton, and R. Wood. Stochastic network interdiction. *Operations Research*, 46(2):184–197, 1998.

[6] G. Dantzig. *On the max flow min cut theorem of networks*, page 227. Stanford University Press, 2003.

[7] E. Dinitz. Algorithm for solution of a problem of maximum flow in networks with power estimation. *Soviet Math. Dokl*, 11(2–4):1277–1280, 1970.

[8] A. Gibbons. *Algorithmic graph theory.* Cambridge University Press, Cambridge, New York, 1985.

[9] T. Harris and F. Ross. Fundamentals of a method for evaluating rail net capacities. Technical Report AD0093458, RAND Corporation, October 1955.

[10] P. Hart, N. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968.

[11] R. Karp. Reducibility among combinatorial problems. In *Complexity of Computer Computations*, pages 85–103. Plenum Press, 1972.

[12] F. Li and G. Kollios. Real Datasets for Spatial Databases: Road Networks & Points of Interest, 2005.

[13] P. Paruchuri, M. Tambe, F. Ordóñez, and S. Kraus. Security in multiagent systems by policy randomization. In *Proc. of AAMAS*, 2006.

[14] J. Pita, M. Jain, M. Tambe, F. Ordonez, S. Kraus, and R. Magori-Cohen. Effective solutions for real-world stackelberg games: When agents must deal with human uncertainties. In *Proc. of AAMAS*, 2009.

[15] J. Tsai, S. Rathi, C. Kiekintveld, F. Ordóñez, and M. Tambe. IRIS-A Tool for Strategic Security Allocation in Transportation Networks. In *Proc. of AAMAS*, pages 37–44, 2009.

[16] A. Washburn and K. Wood. Two-person zero-sum games for network interdiction. *Operations Research*, pages 243–251, 1995.

[17] R. Wood. Deterministic network interdiction. *Math. and Comp. Modelling*, 17(2):1–18, 1993.