Optimizing Implicit Proxy Placement to Evade Traffic Filters

Jacopo Cesareo

Master's Thesis

Presented to the Faculty
of Princeton University
in Candidacy for the Degree
of Master of Science in Engineering

Recommended for Acceptance
by the Department of Computer Science
Princeton University

Adviser: Jennifer Rexford

June 2012

Abstract

Traffic filters block clients from communicating with certain Internet destinations. To prevent clients from evading the filtering policies, traffic filters may also block access to well-known anonymizing proxies. In response, researchers have designed more sophisticated circumvention techniques that rely on *implicit* proxies lying along the path to unfiltered destinations. An implicit proxy transparently deflects traffic directed to an unfiltered destination toward the filtered destination. However, the effectiveness of implicit proxies highly depends on their presence in paths between clients and unfiltered destinations. In this paper we formulate and solve the problem of proxy placement, and evaluate our algorithms on snapshots of the Internet topology for a variety of client and destination sets. We also consider smart filtering techniques that select alternate routes to avoid implicit proxies, as well as the effects of asymmetric Internet routing. Our results show that a relatively small number of proxies can satisfy a large group of clients across a range of geographic locations.

Acknowledgments

I would like to thank my adviser Jennifer Rexford for her invaluable advice and support. Our recurrent meetings and discussions were a true source of inspiration. I would like to thank Josh Karlin and Michael Schapira for their collaboration. Josh, you always provided insightful perspectives to augment our work. Michael, your theory wizardry helped me fully grasp the 'complexity' of the problem at hand.

I would also like to thank the members of the Cabernet Research Group as well as fellow graduate students Matvey Arye and Nicholas Jones for their feedback and comments.

Lastly, I would like to thank Allison and my family, Paola and Ludovica, for their continuous moral support and standing by me every day of my life.

Ai miei nonni.

Contents

1 Introduction

Network service providers increasingly block, filter, redirect, intercept, or even modify traffic between their users and popular or controversial websites or other Internet-based services [2, 7, 24]. Most circumvention techniques [1, 8, 10] rely on *explicit proxies*, where the clients send their packets to a publis VPN server or an anonymizing proxy like TOR (The Onion Router) [10], which in turn directs traffic to the filtered destination. However, service providers can block access to these proxies simply by adding them to the list of filtered IP addresses. This forces the proxy services to change IP addresses (or even IP prefixes) frequently, in an ongoing cat-and-mouse game with the traffic filters.

Recent circumvention techniques based on *implicit* proxies avoid these problems by having the proxy lie passively on the path from the clients to seemingly innocuous destinations. However, the success of these techniques depends on placing the implicit proxies at strategic locations that lie on many paths between clients and unfiltered destinations.

1.1 Decoy Routing to Evade Traffic Filters

Implicit proxies are an effective way to offer services to clients without explicit configuration. Historically, service providers deployed implicit Web proxies at client access points, to serve cached content without requiring users to configure their browsers to use a proxy. Using implicit proxies to circumvent traffic filtering raises additional challenges. First, the proxies must be placed outside of the region controlled by the traffic filter, making it harder to ensure that client traffic traverses the proxy. Second, clients must simultaneously obscure the IP addresses of covert destinations (to evade the traffic filters), and signal the real addresses to the implicit proxy (to ensure the traffic reaches the intended destination).

During the past few years, three works have proposed effective ways to use implicit proxies to circumvent traffic filters: Cirripede [17], Decoy Routing [20] and Telex[25]. Even though there are subtle differences between them, Cirripede, Decoy Routing and Telex share a common use case, shown in Figure 1. A client accesses Internet services through a traffic filter that blocks access to the address of the covert destination. If the client tries to connect to the covert destination explicitly, the connection is blocked by the filter (1). To circumvent the filter, the client initiates a connection to a non-filtered destination address (2). In reality, this connection camouflages a

covert signal with the intent of the client to connect to the implicit proxy. A router on the path of the flow detects the covert signal and redirects the flow to the implicit proxy (perhaps running directly on the router), which in turn directs the traffic to the covert destination (3). Cirripede, Decoy Routing and Telex have given different names to the non-filtered destination as well as the router-proxy combination. Without loss of generality, we use the nomenclature used in Decoy Routing: we refer to *decoy destinations* for non-filtered destinations and *decoy routers* (DR) for the traffic-deflecting components. We refer to the overall scheme as *decoy routing*.

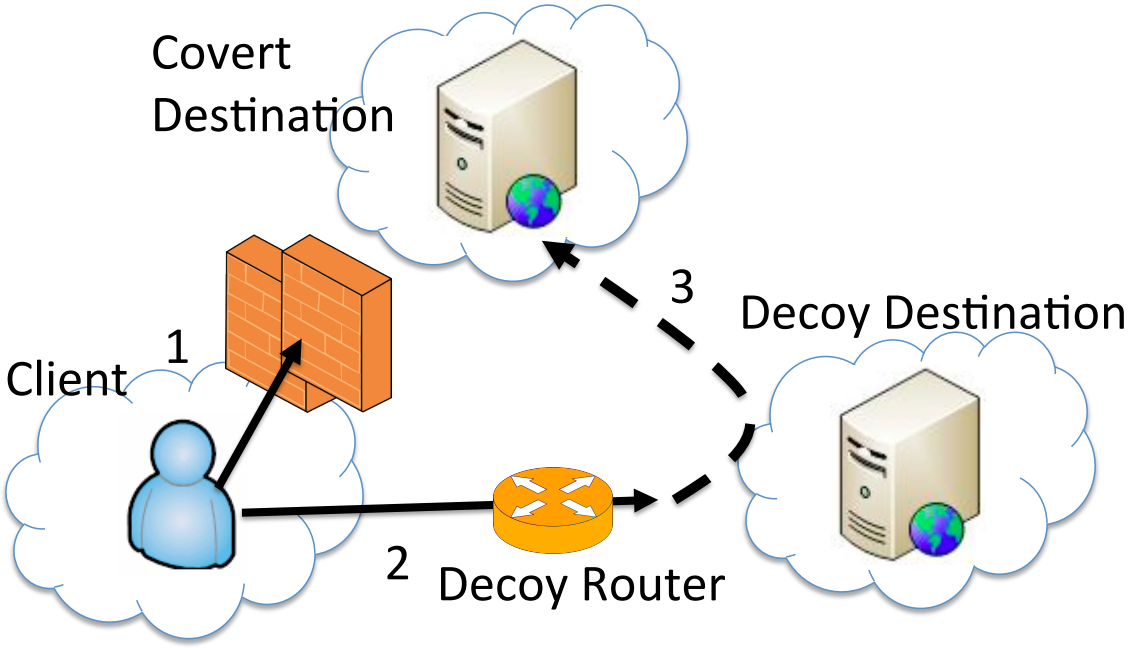

Figure 1: Decoy Routing Scheme

The success of decoy routing rests on two conditions: 1) the traffic filter cannot distinguish the covert signal from legitimate traffic and 2) the DR lies on the path between client and decoy destination. The former condition is solved by injecting pseudo random values in traffic headers.[1]

The latter generates the more complex problem of *Decoy Router Placement*.

[1]In Cirripede the signal is hidden within the initial sequence number of TCP SYN packets. In Decoy Routing and Telex the covert signal is the manipulation of the random nonce in the *Hello* packet in the TLS protocol. Specifically, in Decoy Routing the nonce is a shared secret between client and DR while in Telex it is the DR's public key.

1.2 Optimizing Decoy Router Placement

Since clients do not send packets directly to the decoy router, traffic filters cannot easily block access. To block access, the service provider has to block all traffic that traverses the DR en route to decoy destinations. On the other hand, the DR must lie on the path between the client and the decoy destination. That is, effective DR placement is not just important for good performance—it is crucial for the solution to work in the first place. Placing DRs at many locations throughout the Internet could be prohibitively expensive: the goal is to place them strategically to cover many filtered clients at minimal cost.

The decoy router placement (DRP) problem is the problem of placing decoy routers in the network to maximize the number of filtered clients that traverse DRs en route to decoy destinations. A 'naïve' solution would deploy a decoy router on each path between a client and a decoy destination. A more sophisticated solution would strategically place decoy routers at locations that appear on many paths, to maximize coverage with as few decoy routers as possible.

Previous research in the area has stopped short of exploring the optimal number and distribution of DRs. The work in Cirripede [17] touches on the subject suggesting that two Tier-1 Autonomous Systems (ASes) would need to be instrumented, but many questions remain about how to place decoy routers, and how to handle smart traffic filters that attempt to circumvent the decoy routers.

In this paper we make the following contributions:

- We formulate the DRP problem as a monitor placement problem, and show that the problem is NP hard.
- We present effective heuristics to find efficient DRP solutions and propose two metrics to evaluate them.
- We introduce a more challenging variant of the DRP problem, DRP against *smart* filtering (DRPSF), and show its complexity and inapproximability bounds.
- We evaluate the DRP and DRPSF problems on a wide range of real-world scenarios and show that efficient DR placement is indeed feasible no matter the clients, decoy destinations, filtering level or path properties.

- Lastly, we relate the results of the DRP problem to the structure of the network and conclude that the latter makes efficient placement possible.

2 Decoy Router Placement

In this section we formulate the DRP problem (§2.1). Then, in § 2.2 and § 2.3, we propose two metrics to evaluate candidate solutions, and their respective algorithms for finding such solutions. We also report the theoretical results in support of the relationship between the algorithms' solutions and the optimal solution.

2.1 Decoy Router Placement Problem

We abstract the AS-level Internet as a graph $G = \{V, E\}$ of nodes V and edges E. Nodes $v \in V$ represent Autonomous Systems (ASes), and edges $e \in E$ represent logical connections between them. An AS, or domain, is a collection of devices under the control of a single entity. We model the components of the decoy routing scheme with respect to the AS-level graph G.

Clients and Decoy Destinations Clients and Decoy Destinations are *nodes* in the graph. In particular, we define a set $C \subset V$ of clients and a set $D \subset V$ of decoy destinations. It is worth noting that in our model no node can act as both a client and decoy destination.[2]

Paths Traffic flows from a node i toward node j on *path* p_{ij}. Since traffic at the AS level is often asymmetric, path p_{ji} might not be the same as path p_{ij}. Initially, we make the assumption that there is a *single* path from client i to decoy destination j. We later consider the case where a client can choose to reach the decoy destination through a set of routes (thus enabling a smart filter to avoid the route with a DR). As, for now, every client-destination pair is connected by a single path, we shall use the terms *pair* and *path* interchangeably. We define P as the set of all paths between clients in C and decoy destinations in D.

The Cirripede, Decoy Routing and Telex schemes differ in whether or not their decoy routers need to observe return traffic. If a decoy router is on path p_{ij} (client i to destination j) and p_{ji} then the router has more information and control over the flow. The decoy routers in the

[2] $C \cap D = \emptyset$

Notation	Definition
C	set of Clients
D	set of Decoy Destinations
p_{ij}	Path from $c_i \in C$ to $d_j \in D$
P^i	set of Paths between $c_i \in C$ and D
P	set of Paths between C and D
R	DR Candidate Solution set
N_x	AS neighbor set of AS X
P^i_j	set of all valid paths between c_i and d_j

Table 1: Decoy Router Placement Notation

Telex [25] architecture must observe traffic in both directions. We call this the *bidirectional* requirement. Cirripede and Decoy Routing [17, 20] only need to observe traffic on path p_{ij} (termed *unidirectional*). Clearly, the bidirectional requirement will reduce the number of available decoy destinations provided by each DR, as discussed in § 5.4.

Decoy Routers We associate a DR with either a node or edge in the graph. Thus, our analysis will identify either entire ASes or individual inter-AS links to instrument with DRs. Identifying ASes to cover gives a bound on the number of individual organizations that would be needed to deploy decoy routers. Identifying inter-AS links to instrument is more fine grained and helpful in the event that the decoy router is simply a bump-in-the-wire device instead of an actual router. We define R as the set of candidate DR locations. We assume that a client node will not host DRs.[3]

The DRP problem is therefore equivalent to finding a set of nodes or edges R to **cover** paths in P. The goal is to find a solution R which covers P "efficiently".

We give a specific formulation of the problem: given a fixed number of DRs k, what is the best placement solution R for these DRs? To evaluate a solution we propose two metrics: **fraction of pairs covered** and **fraction of α-covered clients**. We discuss these metrics in the following sections.

For clarity, the notation introduced so far is summarized in Table 1.

2.2 Fraction of Pairs Covered

The first metric we propose, fraction of pairs covered, evaluates the goodness of a solution R of size k by the fraction of client-decoy destination pairs in P covered. The fraction of pairs covered

[3] $R \cap C = \emptyset$

is equivalent to the average success rate of any client c to leverage successfully the decoy routing scheme when picking a decoy destination d at random from set D.

Given this metric, the goal of the the DRP problem is to maximize the fraction of pairs covered in P with a fixed number k of routers R:

$$max_{R:|R|=k}|\{p \in P|\exists r \in R \text{ s.t. } r \in p\}|$$

We refer to this problem as *FPC-k*.

2.2.1 Complexity

We prove the following inapproximability result for *FPC-k*.

Theorem 1. *FPC-k is NP-hard to approximate within a ratio better than* $1 - \frac{1}{e}$.

Proof. To prove the theorem, we show an approximation-preserving reduction from the MAX-K-COVER problem to *FPC-k*. In MAX-K-COVER the input is a universe of elements $U = \{1, \ldots, n\}$ a collection of subsets of the universe $S_1, \ldots, S_m \subseteq U$, and a parameter $1 \leq k \leq m$. The objective is to find the k subsets that cover the maximum number of elements in U. Feige [11] shows that MAX-K-COVER cannot be approximated within a ratio better than $1 - \frac{1}{e}$. We now show a simple approximation-preserving reduction from MAX-K-COVER to *FPC-k*. Given an instance of MAX-K-COVER, construct a network graph as follows. For every element in $e \in U$ create a client vertex c_e and a decoy destination vertex d_e, and for every set S_i create a candidate DR vertex CDR_i. Now, create an edge between every client c_e and destination d_e and every vertex CDR_i such that $e \in S_i$. Observe that every solution to MAX-K-COVER translates to a solution to *FPC-k* with the same value (simply choose the DRs corresponding to the chosen subsets in MAX-K-COVER), and vice versa. Thus, an approximation ratio better than $1 - \frac{1}{e}$ for *FPC-k* would imply an improved approximation ratio for MAX-K-COVER. □

We now present a greedy algorithm with a (tight) $1 - \frac{1}{e}$ approximation ratio, which we will refer to as *GreedyPairs*.

2.2.2 GreedyPairs Algorithm

A greedy algorithm's trademark is to make the locally optimal choice in each of its iterations. For *FPC-k*, the local optimal choice is picking the location covering the largest number of (previously uncovered) $< c, d >$ pairs.

GreedyPairs starts by considering P as the set of *outstanding* paths $OP = P$. Then, it iteratively picks the most popular location, updating set OP accordingly. We summarize *GreedyPairs*'s steps in Alg. 1.

Algorithm 1 GreedyPairs

- Ranking: rank each location based on the number of paths in OP traversing it
- Greedy Choice: pick element x with highest rank (breaking ties arbitrarily). Add x to R
- Input Update : update outstanding paths by removing from OP all paths P_x containing element x.
- Termination: if $|R| = k$, stop. Otherwise, repeat from Step 1

We prove the following.

Theorem 2. GreedyPairs *provides a* $(1 - \frac{1}{e})$ *approximation to the optimal solution in FPC-k.*

Proof. Let $DR \subset V$ be the set of all decoy routers and $CDR \subset DR$ be the candidate solution set for decoy routers. We define the function $f : 2^{CDR} \rightarrow N_+$ that maps every subset $S \subseteq DR$, $f(S)$ to the number of paths in P that traverse at least one element in S. Observe that *FPC-k* is equivalent to finding a set $O \subseteq CDR$ of cardinality k for which the value of $f(O)$ is maximized.

We make the following observations about f. Clearly, f is nondecreasing (as the number of paths covered cannot decrease as more DRs are selected). Also, the following "decreasing marginal contribution" property holds for f:

$$\forall S, T \; s.t. \; S \subset T \subset CDR, \; \forall j \in DR \setminus T,$$

$$f(S \cup \{j\}) - f(S) \geq f(T \cup \{j\}) - f(T)$$

In other words, the marginal utility of adding an element to a set S decreases as the number of elements in S increases. To see this, simply observe that, for every specific $j \in CDR$, the more DRs are chosen the smaller is the set of additional paths that can be covered by adding j to the

set of selected DRs (that is, j's marginal contribution decreases as the set of previously chosen DRs increases).

[21] shows that for functions that are (1) nondecreasing, and (2) have the above decreasing marginal contribution property, a simple greedy algorithm, which translates to *GreedyPairs* in our context, approximates the optimal solution (*i.e.*, the value $f(S)$ across all sets $S \subseteq CDR$ of size k) within a factor of $1 - \frac{1}{e}$. □

2.3 Fraction of α-covered clients

The second metric we propose focuses on *client coverage*. Each client c_i connects to several, if not all, decoy destinations $d \in D$ through a set of paths P^i. $P^i \subset P$ is the set of all paths with source c_i. The key observation is that P^i does not need to be fully covered for a client to leverage decoy routing. Rather, an acceptable fraction α of P^i covered guarantees the client a good chance of leveraging decoy routing. For example, for $\alpha = 0.5$, the client has a fifty percent chance of successfully leveraging the decoy routing scheme when randomly picking a decoy destination. Thus, we consider a client to be covered if α of its pairs are covered.

Through the notion of α-covered clients, we evaluate the goodness of a solution R as the **fraction of α-covered clients** (FαCC). Similarly to §2.2, we can formulate the DRP problem as the problem to maximize the number of α-covered clients given a fixed solution size k:

$$max_{R:|R|=k}|\{c \in C|\text{C is } \alpha - \text{covered by R}\}|$$

We refer to this problem as *FαCC-k*.

2.3.1 Complexity

We note that an approximation-preserving reduction from the DENSEST-SUBGRAPH problem [12, 15] to *FαCC-k* (when $\alpha = 1$) gives strong evidence that, unlike with *FPC-k*, no constant approximation for this problem exists (as, to date, no constant-approximation algorithm for DENSEST-SUBGRAPH was found). We present a variation of the greedy algorithm presented in §2.2 for this problem which fares well in practice.

2.3.2 GreedyPairsPercetage Algorithm

We slightly change *GreedyPairs* as follows. *GreedyPairsPercentage*, adds a fifth step to each iteration of *GreedyPairs* to account for clients that have already been α-covered. *GreedyPairsPercentage*'s steps are summarized in Alg. 2:

Algorithm 2 GreedyPairsPercentage

- Ranking: rank each location based on the number of paths in OP traversing it
- Greedy Choice: pick element x with highest rank (breaking ties arbitrarily).
- Input Update : update outstanding paths by removing from P all paths P_x containing element x.
- **Client Update: remove all paths $p \in P^i$ from OP if α paths in P^ihave been covered**
- Termination: if $|R| = k$, stop. Otherwise , repeat from Step 1

As stated in the previous section, the solution R produced by *GreedyPairsPercentage* does not have provable approximation guarantees. In other words, we do not know what kind of relation it has to the ideal optimal solution. Yet, it is of empirical value and a vehicle for comparison against *GreedyPairs*.

3 DRP Against Smart Filtering

The Internet is a dynamic space and BGP [22], the inter-domain routing protocol, allows an AS to change paths for an IP prefix over time. Without digging deep into the mechanisms of inter-domain routing, let us note that an AS X can obtain multiple paths to the same prefix. These *valid* paths are distributed to X by its neighbors.

In terms of the decoy routing scheme, a client could have more than one path to a decoy destination. Having more than one path to the same decoy destination gives a filtering entity, *e.g.* the ISP, a chance to circumvent decoy routing.

Let's assume that the filtering entity can discover the presence of a DR on one of its paths.[4] Once the DR-covered path is discovered, the filterer can choose a DR-free path to the same decoy destination to circumvent the DR deployment. An example scenario is shown in Figure 2.

We refer to this new problem as *decoy router placement against smart filtering* (DRPSF). In

[4]For example, by obtaining a copy of the decoy routing software and acting as a client.

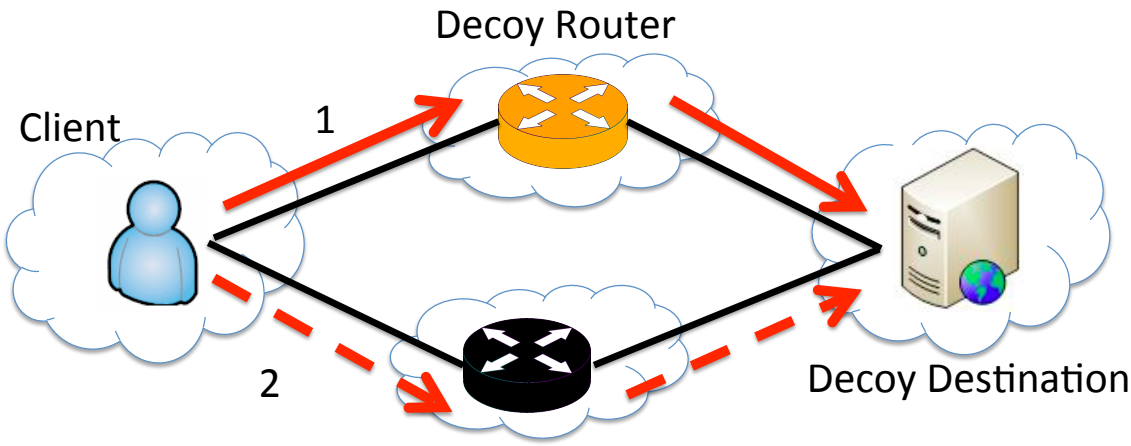

Figure 2: A DR deployment covers the path between a client and a decoy destination (1). A *smart* filtering entity discovers the DR and chooses a neighboring path to reach the decoy destination, thereby circumventing the DR (2).

DRPSF, an effective candidate solution R covers every valid path between a $< c, d >$ pair. In the following sections we formally define DRPSF, analyze its complexity and propose an algorithm to calculate candidate solutions sets.

3.1 DRPSF Problem Formulation

To analyze DRPSF, we need to introduce the notation for AS neighbors. In a graph $G = \{V, E\}$, $v_x \in V$ *neighbors* $v_y \in V \iff \exists e \in E | v_x \in e, v_y \in e$. We denote the set of AS_x's neighbors as N_x. As stated previously, neighbors are important because they provide filtering entities leeway to circumven the decoy routing scheme.

The introduction of multiple valid paths between a client and a decoy destination identifies a client-destination pair $< c, d >$ with a set of paths P_j^i rather than a single path p_{ij} (§2.1). Formally, P_j^i is composed of the path p_{ij} plus all feasible paths from N_i to d_j:

$$P_j^i : p_{ij} \cup p_{xj} | x \in N_i \text{ and } \exists p_{ix}$$

The newly introduced notation is summarized in Table 1.

Overall DRPSF has introduced two changes: the number of paths to cover in P has increased due to neighboring paths and each $< c, d >$ pair now has more than one path. Nevertheless, the metrics to evaluate DRP, (§ 2.1) are still valid for DRPSF.

3.2 Complexity

From a complexity standpoint, losing the single path-per-pair assumption means the approximation guarantees of Theorem 2 no longer hold. Thus, *GreedyPairs* no longer provides a constant approximation to the optimal solution with respect to *FPC-k* (§ 2.2).

Posed from a different perspective, though, both the DRP and DRPSF problems relate to an interesting theoretical bound. We refer to *min-k* as the problem of finding the *minimum* number of DRs to cover *every path* between clients and decoy destinations. Unlike $FPC - k$, *min-k* does not measure partial coverage of pairs, but rather the number of resources used to achieve total coverage. With respect to the *min-k* objective, DRP and DRPSF are similar from a theoretical perspective.

We prove the following inapproximability result for *min-k*.

Theorem 3. *min-k is NP-hard to approximate within a ratio better than* $\ln(|P|)$.

Proof. To prove the theorem, we show an approximation-preserving reduction from the SET-COVER problem to *min-k*. In SET-COVER the input is a universe of elements $U = \{1, \ldots, n\}$ a collection of subsets of the universe $S_1, \ldots, S_m \subseteq U$. The objective is to find the minimum number of subsets needed to cover all elements in U. Feige [11] shows that SET-COVER cannot be approximated within a ratio better than $\ln(n)$. We now show a simple approximation-preserving reduction from SET-COVER to *min-k*. Given an instance of SET-COVER, construct a network graph as follows. For every element in $e \in U$ create a client vertex c_e and a decoy destination vertex d_e, and for every set S_i create a candidate DR vertex CDR_i. Now, create an edge between every client c_e and destination d_e and every vertex CDR_i such that $e \in S_i$. Observe that every solution to SET-COVER translates to a solution to *min-k* with the same value (simply choose the DRs corresponding to the chosen subsets in SET-COVER), and vice versa. Thus, an approximation ratio better than $\ln(|P|)$ for *min-k* would imply an improved approximation ratio for SET-COVER. □

We now show that by slightly changing *GreedyPairs* it is possible to achieve a (tight) $\ln(|P|)$ approximation ratio.

3.3 GreedyPairs for Min-k Algorithm

To solve for *min-k* the only alteration we have to apply to *GreedyPairs* is in its termination condition. Specifically, *GreedyPairs* does not terminate after k iterations, but only after OP is empty. Below we report the changes to the algorithm *GreedyPairs* to match the *min-k* formulation:

Algorithm 3 GreedyPairs for Min-K

- Ranking: rank each location based on the number of paths in OP traversing it
- Greedy Choice: pick element x with highest rank (breaking ties arbitrarily). Add x to R
- Input Update : update outstanding paths by removing from OP all paths P_x containing element x.
- **Termination: if OP is empty, stop. Otherwise , repeat from Step 1**

This modified *GreedyPairs* algorithm has the following approximability guarantee:

Theorem 4. GreedyPairs *provides a* $\ln(|P|)$ *approximation to the optimal solution in min-k.*

Proof. To prove the theorem, we show an approximation-preserving reduction from *min-k* to SET-COVER (using similar arguments to those used in the proof of our inapproximability result above) and observe that the greedy $\ln(n)$-approximation algorithm for SET-COVER translates to *Greedy-Pairs* in our context. □

Thus, when solving for *min-k GreedyPairs* returns a solution which is logarithmic in the input to the optimal solution, no matter if we are in a regular DRP setting or a more complex DRPSF one.

4 Evaluation Framework

In this section we describe the framework used to support the analysis of the DRP problem. We start by presenting the data used for the analysis in §4.1. All the datasets are publicly available at CAIDA [4]. Then, in §4.2 we discuss the step that precedes the analysis: how we generate the paths P that are the input to a DRP problem. Lastly, in §4.3 we discuss the core of the analysis: the execution of the algorithms on the paths generated in the previous step.

4.1 AS-Level Topology and Routing Policies

We use CAIDA's dataset on AS relationships [6] to construct the AS network graph. The dataset is a weekly updated archive of *business relationships* between ASes in the network. Relationships are important because they dictate the flow of traffic on the Internet: an AS establishes its *routing policies* based on relationships with its neighbors. Thus, a path between any pair of ASes in the network derives from a valid composition of these relationships.

Unfortunately these relationships are not publicly available. To generate such a dataset, CAIDA first collects data[5] from geographically and topogically diverse locations, applies a relationship-inference algorithm and lastly validates them through cross-comparison.[9] The algorithm used for inference maintains a conservative approach to avoid incorrect results. Thus, not every relationship is present in the dataset. The datasets used for the analysis contain over 150000 relationships to model approximately 50000 distinct ASes.

We also use CAIDA's dataset on AS information to match each AS to its country [5] so we can evaluate DR placement for clients and decoy routers in various countries.

4.2 AS-Path Generation

To evaluate a DRP algorithm, we first need to generate the paths P between clients and decoy destinations. We use the *routing tree algorithm* specified in Goldberg's *et al.* ([14]) to generate such paths.

The algorithm models the flow of traffic by evaluating relationships between neighboring ASes. Starting from a root AS, the decoy destination in our case, relationships with neighbors are considered in a specific order to maintain network stability conditions [13]. Specifically, the algorithm first takes into account customer-to-provider relationships. A customer AS conveys traffic to its provider AS for any destination. The second stage adds to the set of valid links peer-to-peer relationships: one hop links between ASes that convey each other's traffic. The last stage adds provider-to-customer links. At completion, the tree of links represents paths between every AS in the network and the root AS. We query the tree for paths between clients (nodes in the tree) and the decoy destination (the root of the tree) and add these to P. We generate a tree for each decoy destination to obtain all paths between all clients and decoy destinations.

[5]BGP tables snapshots from Routeviews [3] servers

Noteworthy is that this process only generates paths from clients to decoy destinations. Where both directions are required (*i.e.* , to study Telex in §5.4), the path from the decoy destination to the client is generated by running routing-tree algorithm with clients as the root of the tree.

Overall, a set P can range in size from a few hundred paths to several millions of paths. The size depends on the sizes of sets C and D as well as on the amount of information present in the original relationship dataset.

4.3 DRP Algorithm Execution

The core part of the analysis is the execution of the algorithms proposed in § 2.2.2 and § 2.3.2 on the set P generated in the previous step. The algorithms start by scanning the files containing P and storing their states: information on paths, pairs and clients. The appropriate data structures are built to facilitate the successive iterations of the algorithm; in particular, mappings between nodes/edges and pairs, and pairs and clients, as well as the ranking for each node/edge are kept. Once the entire set has been parsed, the algorithms start iterating to find the candidate solution. In each iteration a candidate node/edge location is chosen by scanning the rankings. The mappings are then updated accordingly.

Running the algorithms on very large data sets (10-15 GB) requires large amounts of memory. To meet those memory requirements we used the cloud. We run the analysis on the extra large VM instances of Amazon EC2 that have up to 34 GBs of RAM memory.

5 Results

We define a DRP problem as a function of six variables *DRP(C,D,P,M,F,S)*: clients, decoy destinations, paths, metrics, filtering levels and solution types.

Clients C : We associate clients to countries. In other words, a set of clients is the set of all ASes in a specific country. The matching is based on CAIDA's AS information dataset [5]. Associating a set of clients with a country is appropriate because traffic filtering is often dictated by government entities. We examine several countries of various sizes and geo-locations.

Decoy Destinations D : We choose three decoy destination sets of different sizes and properties: ROW, U.S.A. and E-commerce. ROW (*rest-of-the-world*) considers every AS outside the client set C as a decoy destination. This is the ideal set for any DR scheme because everything is a

potential decoy. U.S.A. is the decoy destination set of all ASes in the United States. It represents a large fraction of popular destinations on the Internet. E-commerce represents a small set of popular web commerce sites. Decoy Routing and Telex [20, 25] leverage the TLS protocol, used in e-commerce transactions.

Paths P : We include different path properties to accomodate for different implementations of the Decoy Routing architecture. Decoy Routing [20] and Cirripede [17] only need DRs to be placed on the forward path between the client and decoy destination. Thus, we only need to consider paths in one direction (*unidirectional* paths). Telex, instead, needs a DR to be present on both the forward and return path. We defined this requirement as the *bidirectional* paths requirement in § 2.1.

Metrics M : We have proposed two algorithms to optimize the metrics we have defined in §2.2 and 2.3. We evaluate the solution of a problem by applying a specific algorithm and measuring the "goodness" of the solution proposed through its respective metric.

Filtering Level F : In § 3 we have labeled a filtering entity's attempt to work around decoy routing as **smart** filtering and defined the DRPSF problem. We refer to the original DRP problem, where no attempt to work around decoy routing is made by the filtering entity, as *naïve* filtering.

Solution Type S : We consider two strategies for picking candidate solutions: one focuses on nodes in the AS-level graph, the other on edges.

Clients C	China Egypt Germany Italy Iran Libya Russia Spain Syria
Decoy Destinations D	**E-commerce U.S.A. ROW**
Paths P	**Unidirectional Bidirectional**
Metrics M	**FPC FαCC**
Filtering F	**Naïve Smart**
Solution Type S	**Nodes Edges**

Table 2: Decoy Router Placement Problem Variables

The values for each variable are summarized in Table 2.

In the following sections we analyze the DRP problem by focusing on each variable. The approach we follow is to define a specific **scenario** as a combination of the six DRP variables and to compare results when one or two variables change. The goal is to receive a wider spectrum of results from which to draw conclusions and to highlight the importance of each variable in the problem. We then tune multiple parameters at once to get a better understanding of how many

decoy routers might be required in extreme scenarios.

5.1 Decoy Routers on Nodes vs. Edges

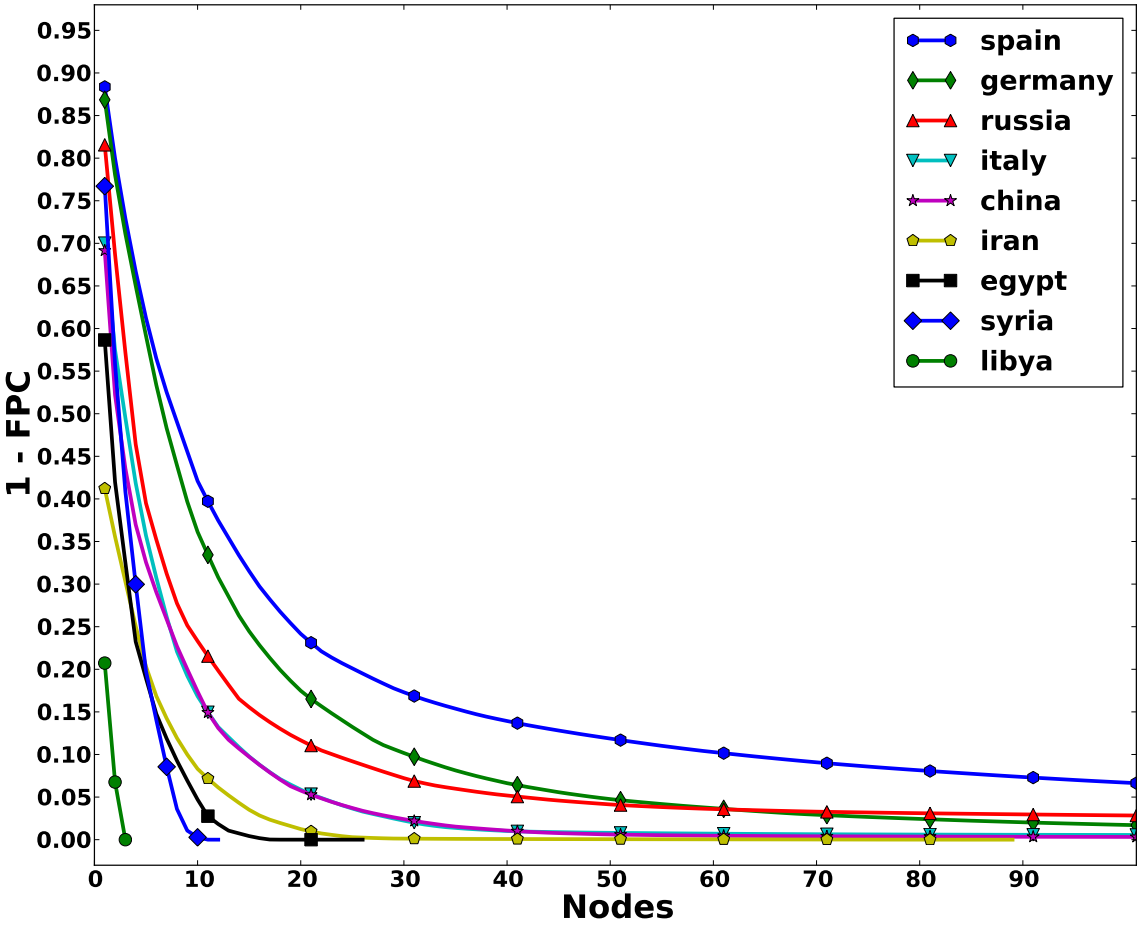

Figure 3: Nodes analysis across different countries

We start by comparing solution comprised of nodes as opposed to edges. Picking nodes provides a coarser analysis: for each picked node we assume we can intercept all traffic traversing the AS.[6] Picking edges provides a more fine grained analysis as we restrict the location of the DR down to inter-AS links, but can prove to be less efficient if traffic is equally distributed amongst many edges. We study the two solution types across every country from which we have collected data and set the decoy destinations set to be the largest possible (ROW). We consider only paths in the forward direction (unidirectional) and a naïve filtering level. We apply the *GreedyPairs* algorithm

[6]This is achieved by instrumeting all of its ingress-egress routers.

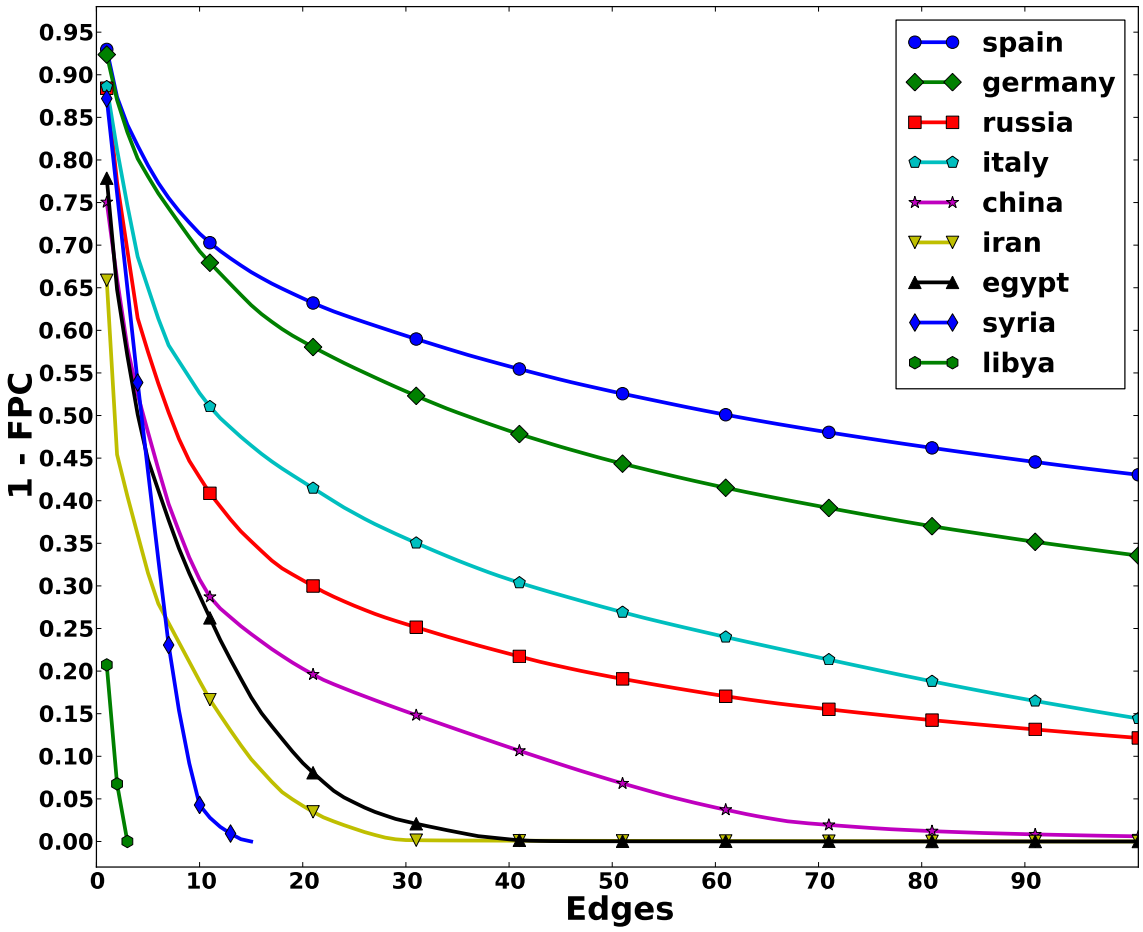

Figure 4: Edges analysis across different countries

and evaluate the solution by metric FPC(§ 2.2).Results are shown in Figures 5.1 and 5.1. The two figures plot the Complementary Cumulative Distribution Function (CCDF) of the fraction of pairs covered. In other words, the fraction of pairs still to be covered given a number of nodes or edges.

Both graphs show curves with similar behavior: an exponential decrease followed by an asymptote. The exponential decrease explains that a small set of elements (either nodes or edges) cover the majority of pairs. Adding these 'popular' locations to the solution is of tremendous value. After a given number of elements, the marginal value of adding more becomes negligible (asymptote). This 'break point' can be considered as the optimal trade-off between pair coverage and resources used.

Table 3 shows the fraction of pairs covered with a fixed number of elements k, as per problem *FPC-k* § 2.2. The value k is picked to the right of every break point in each curve. In the case

Country	Nodes	Edges
spain	0.82	0.40
germany	0.89	0.47
russia	0.92	0.74
italy	0.97	0.64
china	0.95	0.84
iran	0.99	0.99
egypt	1.0	0.97
syria	1.0	0.99
libya	1.0	1.0

Table 3: Fraction of $< c, d >$ pairs covered with 30 elements

of edges, the fraction of paths covered with a fixed number of elements is lower with respect to nodes.

We acknowledge the fact that placing a DR on one inter-AS link is different from placing DRs on an entire AS, yet concentrating DRs on a small number of specific locations may prove easier than distributing DRs across distant and disparate locations. Furthermore, even for larger countries, *i.e.* , Spain Germany Italy and Russia, that present a larger number of ASes and therefore higher number of paths, concentrating DRs on a few nodes mantains a high coverage value. Thus, we conclude the following: 1) 'popular' nodes receive decoy routing traffic from several links and this explains the gap in coverage between nodes and edges and 2) focusing on a small number of specific locations achieves high coverage no matter the geo-location of the client set. We suspect the high coverage achieved regardless of the client set is due to the fact that the 'popular' nodes are large ASes, probably Tier-1 ASes. We investigate this matter further in the following sections.

From this point forward we will continue our analysis by focusing on choosing nodes.

5.2 Locations of Decoy Destinations

Ideally, we would like for every AS outside of the client set to be a potential decoy destination. This would make filtering very hard if not impossible. On the other hand, choosing a small set of decoys could scale the deployment of DRs down orders of magnitude and still prove to be detrimental to the filtering entity's interests. Figure 5 shows the CCDF across different decoy destination sets. Each data point in the curves represents the median value over all countries from which we collected data. The horizontal red line marks 90% coverage.

The scenario variables used are equal to the ones used in the previous section.

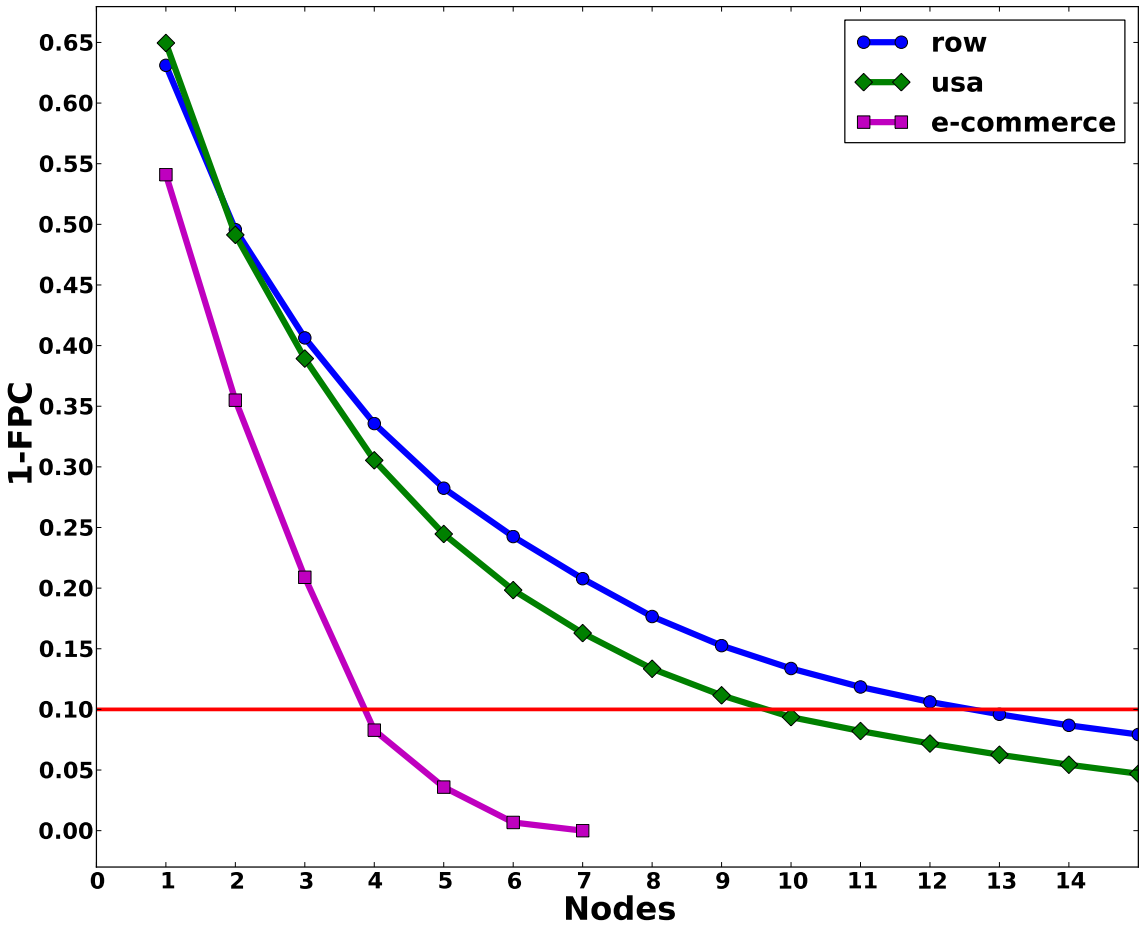

Figure 5: Comparison between solutions proposed by GreedyPairs for different decoy destinations sets.

Results show an almost insignificant growth in the number of nodes needed to be deployed for very differently sized decoy destination sets. In fact, the E-commerce set is comprised of four ASes and needs a number of nodes linear with the decoy destination set to be covered. Instead, U.S.A. and ROW sets have more than 15000 ASes and only need around ten and thirteen decoy routers respectively to achieve the same fraction of pairs covered. Choosing a small number of decoy destinations makes the deployment of DRs possible and even trivial: equip the decoy destination themselves with DRs. The filtering entity, though, may choose to completely block access to the small set of decoy destinations without much collateral damage. On the other hand, having everything outside the client set be a possible decoy leaves the filtering entity no option but disconnect the client AS set entirely from the Internet. Since this option comes with a relatively bigger yet not prohibitive DR deployment, we believe it is desirable to consider large

decoy destination sets for real DR deployments.

5.3 Naïve vs. Smart Filtering

In §3 we introduced the DRPSF problem as the variation of the DRP problem when facing a proactive, smart filtering entity. In this part of the analysis we assess the differences between solutions proposed when applying the *GreedyPairs* against a naïve and smart adversary. It is worth remembering that the main difference between DRP and DRPSF is that paths P in DRP are agumented by valid neighboring paths in DRPSF's P'. Furthermore, a $< c, d >$ pair is now described by more than one path in the DRPSF problem.

The scenario considered is the following: we set the decoy destination set to ROW, consider a pair covered if the forward path contains a DR, and apply *GreedyPairs* to P and P'. We evaluate the solutions through metric FPC and plot the respective CCDFs.

Results are shown in Figure 6. The curves show the median values across all countries.

The smart filtering curve exhibits the same behavior as the naive one: an exponential decrease followed by an asymptote. Thus, even with a higher level of filtering, picking a small number of popular locations accounts for coverage of the majority of pairs.

The difference between the two curves is that the smart filtering one presents lower coverage values with respect to the naive curve. This difference is a constant value ˜10 %. Our interpretation is that this is because most of the paths in P' that are not present in P, *i.e.* paths from clients' neighbors to the decoy destinations, intersect the original paths, *i.e.* the paths in P, somewhere along the way. Therefore, having twice as many paths in P' as in P only slightly affects the overall results.

Our interpreation is that the intersection happens at locations where traffic from different ASes converges to, *e.g.* in the backbone Tier-1 ASes.

5.4 Unidirectional vs. Bidirectional Paths

In § 2.1 we described the different requirements the implementations of Decoy Routing, Cirripede and Telex [17, 20, 25] have in terms of DR placement. In particular, Decoy Routing and Cirripede only need the DR to lie on the forward path between a client and decoy destination. Telex requires the DR to lie on both the forward and return path. We defined these two different requirements the

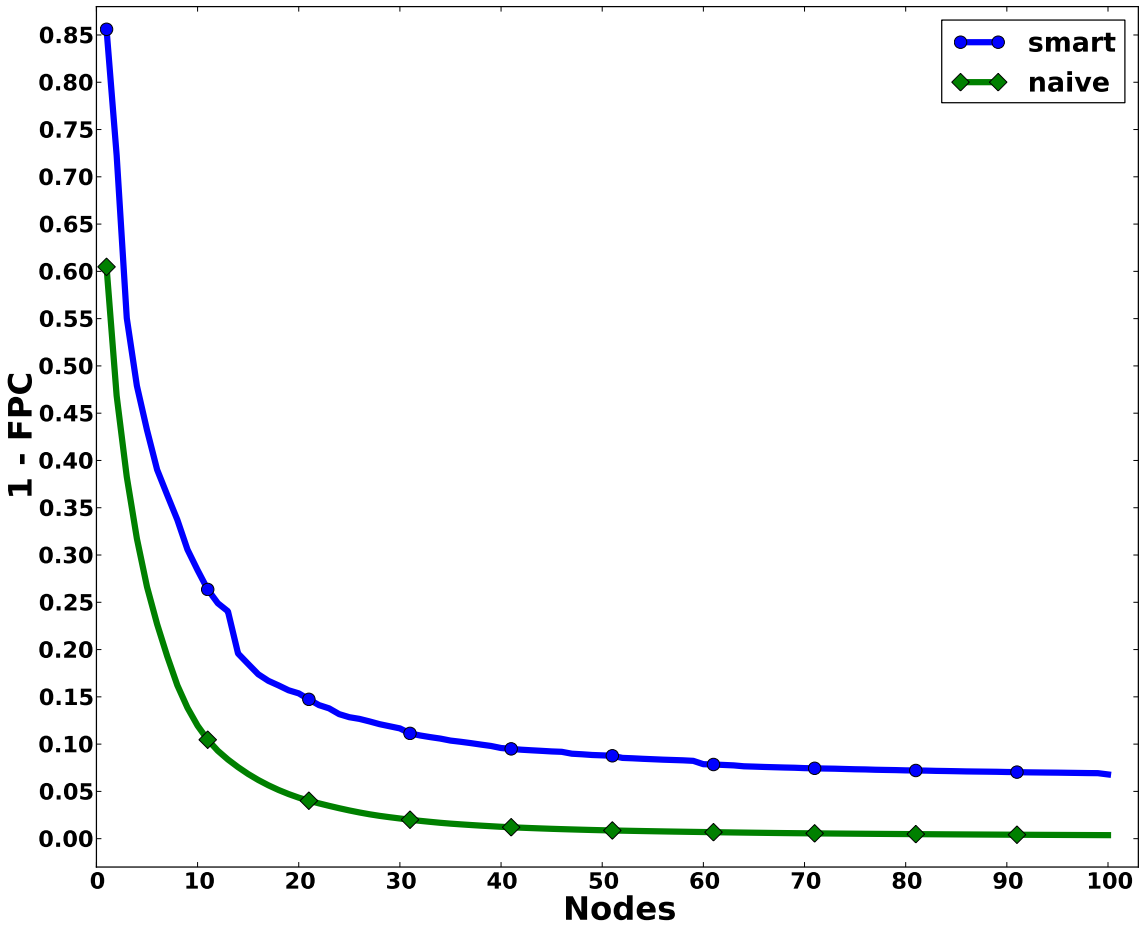

Figure 6: Comparison between naïve (DRP) and smart filtering (DRPSF) problem solutions proposed by algorithm GreedyPairs.

unidirectional and *bidirectional* paths requirements. In the bidirectional paths case, the forward and return path might not coincide. In fact, BGP's flexibility allows ASes to make asymmetric routing decisions to favor their own economic benefits. Therefore, we calculate paths from C to D as well as D to C. To cover $< c, d >$ we can only choose a node or edge present in both paths.

We consider a scenario similar to the ones in the previous sections: decoy destination set is ROW and filtering is naïve.

Figure 7 shows the CCDF of metric FPC metric for both bidirectional and unidirectional paths. Each curve is the median values across all countries considered.

The bidirectional paths curve shows lower values of pair coverage (higher CCDF values) with respect to the unidirectional paths curve. The asymmetry of Internet routing makes it hard for forward and return paths to intersect anywhere but at the decoy destination. This property of the

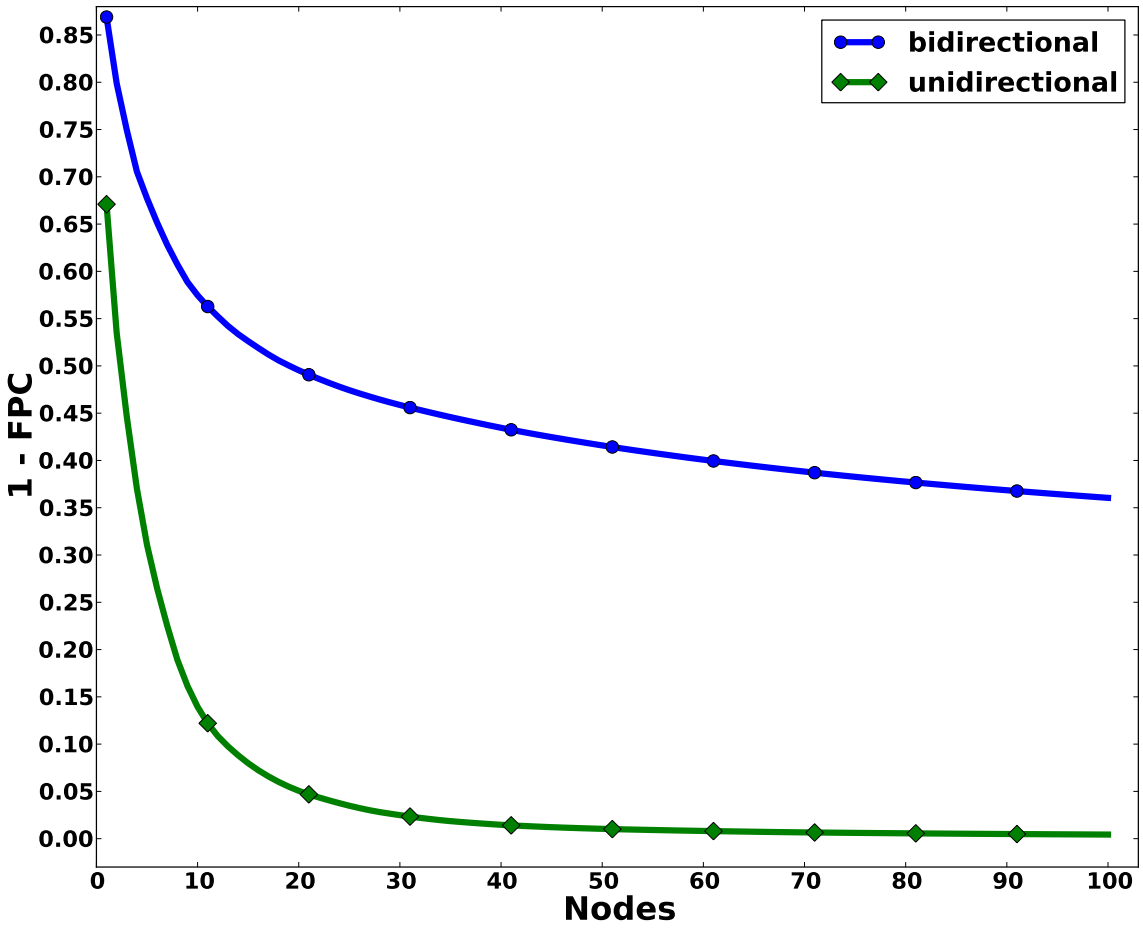

Figure 7: Unidirectional - Bidirectional Paths Comparison

network offers advantages to the Cirripede and Decoy Routing implementation which only need to lie on the forward direction. It seems that a Telex deployment would prove to be less efficient and largely restrict the number of candidate locations.

Noteworthy is that we tried to evaluate the impact of bidirectional paths on an edge deployment. The dataset showed that the majority forward and return paths shared no common edges, and it was impossible to produce meaningful results.

Nevertheless, the bidirectional curve is similar to those we have seen in previous analyses: a small set of popular locations is sufficient to cover the majority of pairs. We conclude that these popular locations are such even if the source and destination nodes are very different. We reiterate our belief that such locations are found in the backbone of the network and associate them with Tier-1 ASes.

5.5 Coverage Metrics

So far in our analysis we have only applied the algorithm *GreedyPairs* and evaluated solutions by metric FPC. We have also proposed a metric centered around client-coverage, FαCC, and an algorithm, *GreedyPairsPercentage*, to optimize solutions for this metric. *GreedyPairsPercentage* does not have the approximability bounds of *GreedyPairs*, yet it might prove worthy of finding more efficient DRP problem solutions. In this part of the analysis we will compare the two algorithms and evaluate solutions found by *GreedyPairs* and *GreedyPairsPercentage* through metric FαCC (§2.3). In other words, we test *GreedyPairs*'s value for client coverage.

We consider two very similar scenarios. In both, the set of decoy destinations considered is ROW and the filtering level is smart. The first scenario, though, accounts for unidirectional paths while the second assumes bidirectional paths. The results for both scenarios are shown in Figure 8.

In the figure, *GreedyPairs* is labeled as 'gp' and *GreedyPairsPercentage* as 'gpp'. The solid lines are the curves for the unidirectional paths scenario. As we can see, there is not a significant difference between them. Thus, *GreedyPairs* does well at α-covering clients. The interpretation is that paths from a client to all decoy destinations, represented by set P^i, are similar among each other. In conjunction with the results from § 5.3, we conclude that a client does not only reach one decoy destination through similar paths, but reaches also different decoy destinations through similar paths.

The dashed curves refer to the scenario with bidirectional paths. These two curves show, surprinsingly, a considerable difference between *GreedyPairs* and *GreedyPairsPercentage*. In fact, *GreedyPairs* outperforms *GreedyPairsPercentage*. The requirement of DR deployment at the intersection of forward and return paths forces the algorithms to pick 'popular' locations. Picking 'popular' locations is the strategy followed by *GreedyPairs*. *GreedyPairsPercentage*, instead, preemptively excludes paths if a client is α-covered. This exclusion, we believe, decreases the popularity of certain locations and induces the greedy choice to be less effective in later stages of the algorithm.

In conclusion, we believe the popularity of locations is heavily skewed in the network. This distinguishes a set of locations that are traversed by the majority of paths, *i.e.* Tier-1 ASes, from others which are traversed by a small number of paths, *i.e.* stub ASes. If we artificially take that

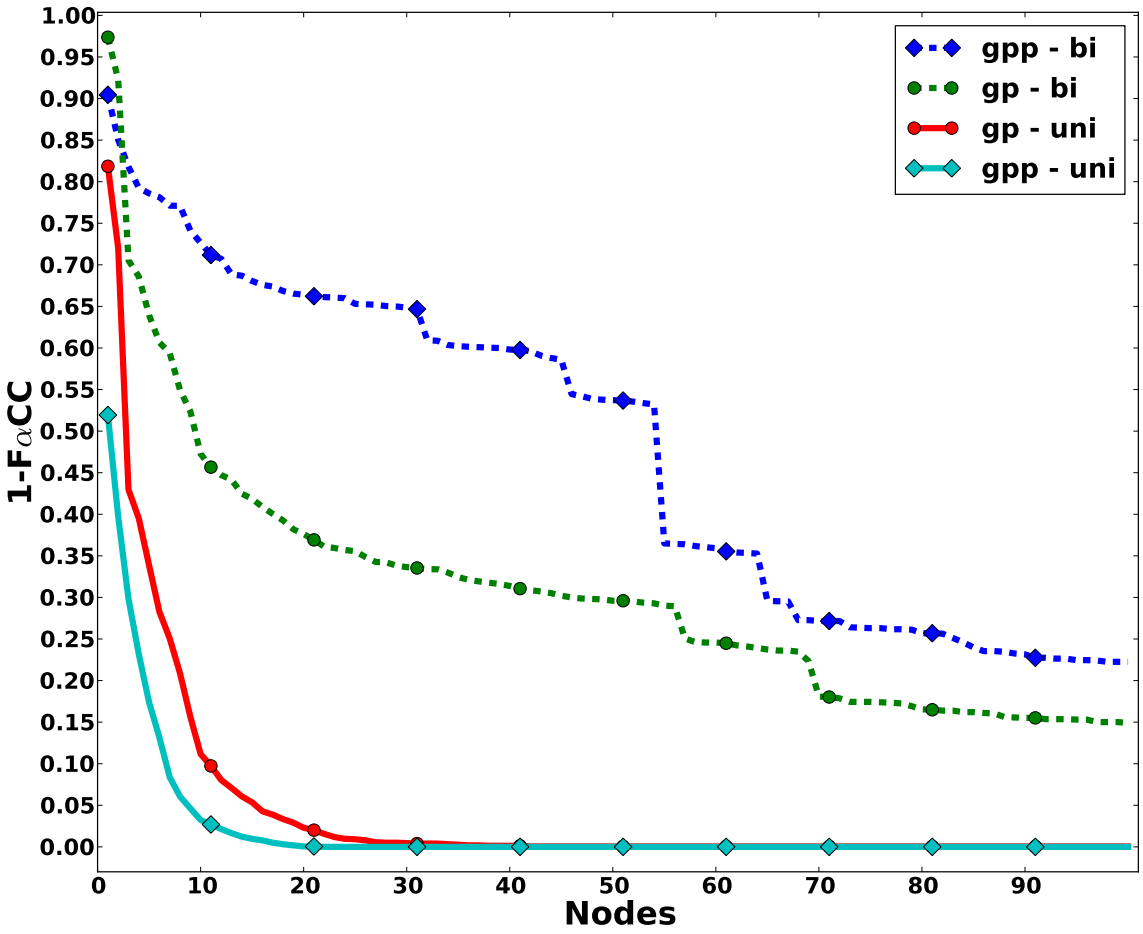

Figure 8: Comparison between algorithms GreedyPairs - GreedyPairsPercentage over metric FαCC.

skewedness away, we inadvertedly also exclude popular locations from our solution.

5.6 Variable Interaction

We have studied how changing a single variable affects the required deployment of decoy routers. To fully understand how the variables interact would require an enumeration of all possible parameters. Instead, we consider two scenarios that could be referred to as the 'best and 'worst' possible case. We believe that the results of the unstudied scenarios should be within these two sets, or at least close to them.

In both scenarios we use the FPC metric and use the ROW destination sets. The remaining parameters are chosen based upon their performance in the individual analysis. The best case scenario instruments ASes instead of edges, assumes that the decoy routing scheme only needs to

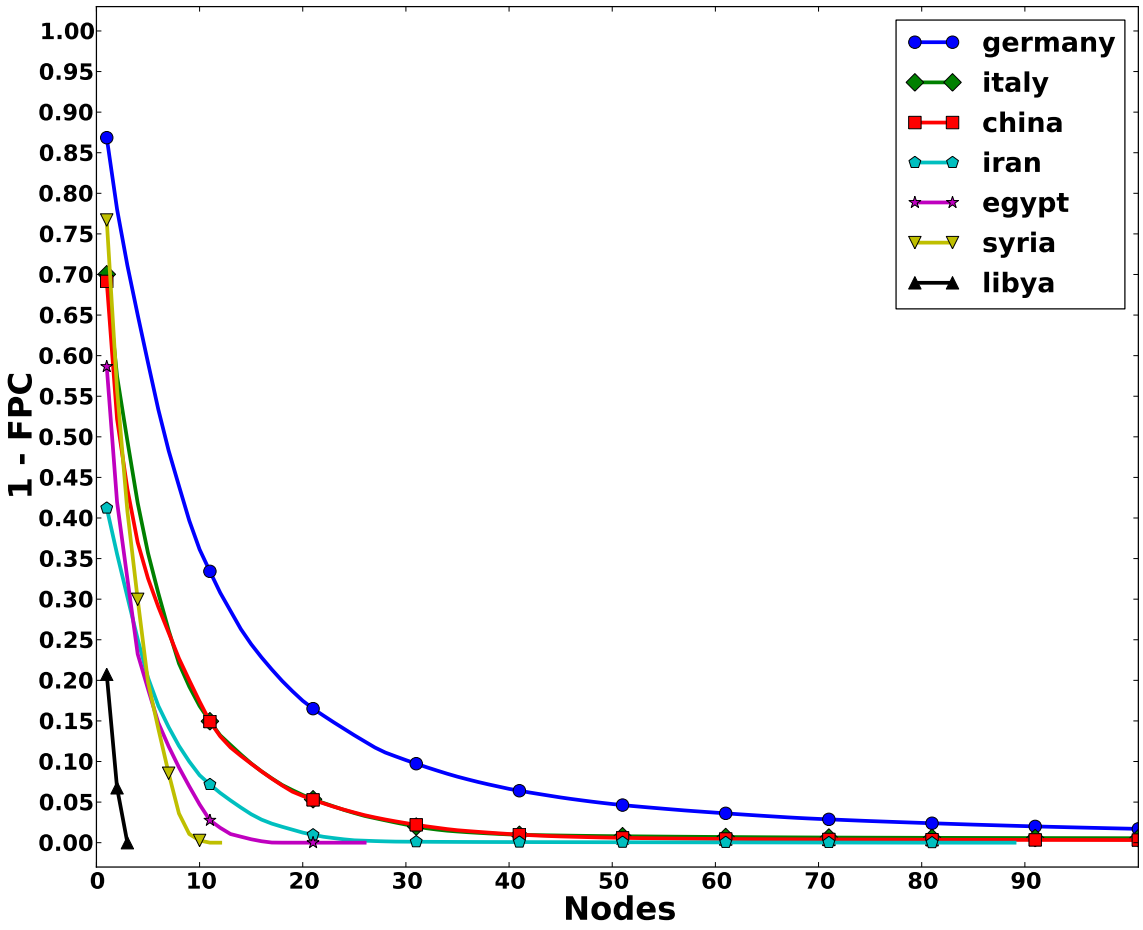

Figure 9: Best case analysis across different countries. The best case is a node deployment over unidirectional paths and with a naïve filtering level.

observe paths in the $< c, d >$ direction (unidirectional), and further assumes that the adversary is unable to adjust its paths to avoid decoy routers. The worst case scenario is the opposite.

The best case scenario is depicted in Figure 5.6 and the worst case is shown in Figure 5.6. The difference is fairly dramatic. As we have seen in previous sections a small number of nodes can provide substantial coverage of the measured countries in the best case. However, in the worst case it appears that a substantial deployment would be necessary to cover significant fractions of many countries.

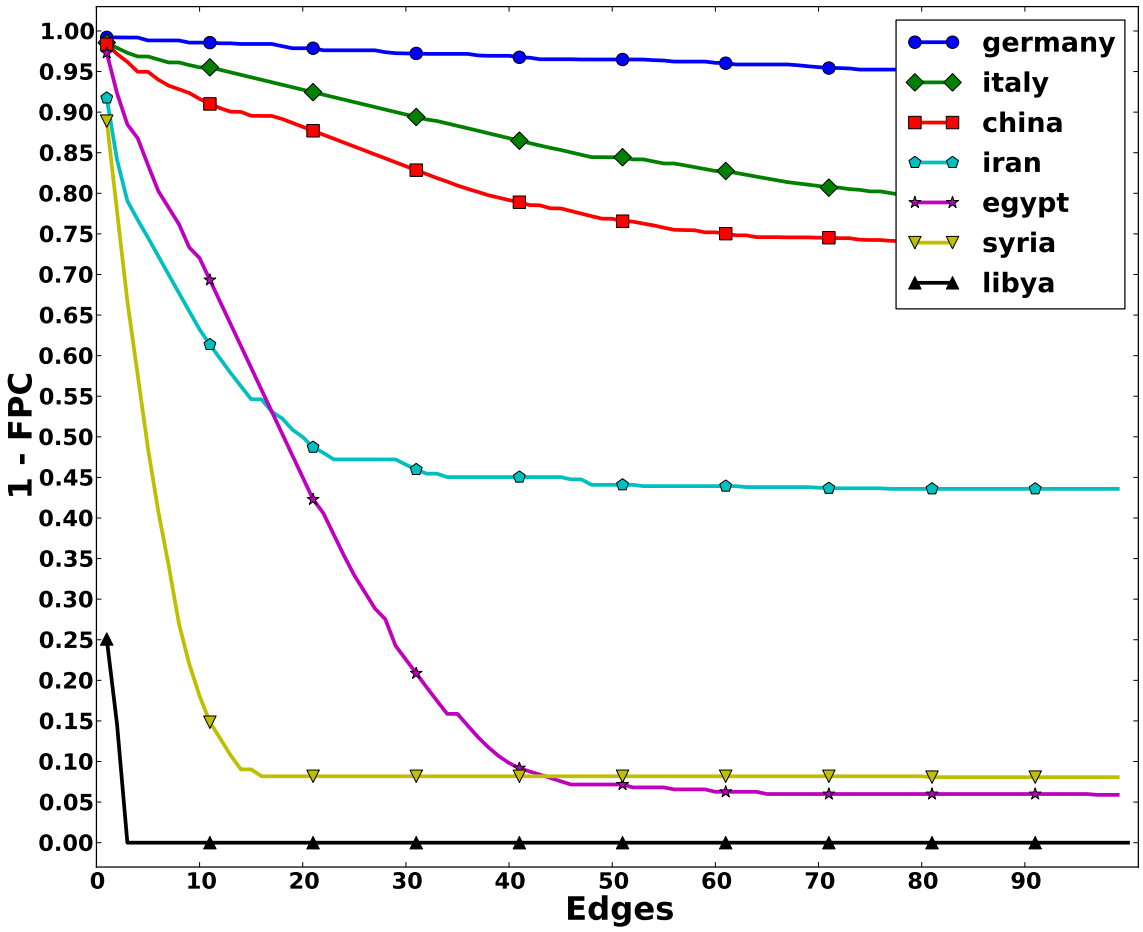

Figure 10: Worst case analysis across different countries. The worst case is an edge deployment over bidirectional paths and with a smart filtering level.

5.7 Decoy Routers Locations

We have given a wide range of perspectives on the DRP problem by analyzing each of its variables and how these effect the solutions proposed by algorithms *GreedyPairs* and *GreedyPairsPercentage*. In the interpretation of the results, we have shown that the DRP problem can be efficiently solved with a small set of decoy routers placed at very popular locations. We have also associated these popular locations with large ASes, *i.e.* Tier-1 ASes.

To confirm such statements, we dedicate the last part of the analysis to answer the following question: where in the network is the best location for decoy routers? We approach this question by verifying the locations of the ASes contained in the solution sets proposed by our algorithms. We distinguish between two positions on each path: close to the endpoints, *e.g.* client and decoy

destination, or in the 'middle' of the path. We associate the middle of the path with positions 'deeper' in the network, where we find Tier-1 and Tier-1 ASes.

We use the following approach: we take the solutions proposed by the scenarios addressed in earlier sections and analyze where on each path the candidate locations are situated. Specifically, we consider all the countries from which we have collected data and the decoy destination set ROW together with different combinations of the variables for path properties and filtering level. Figure 11 shows the results of our analysis. We have four combinations between the variable for

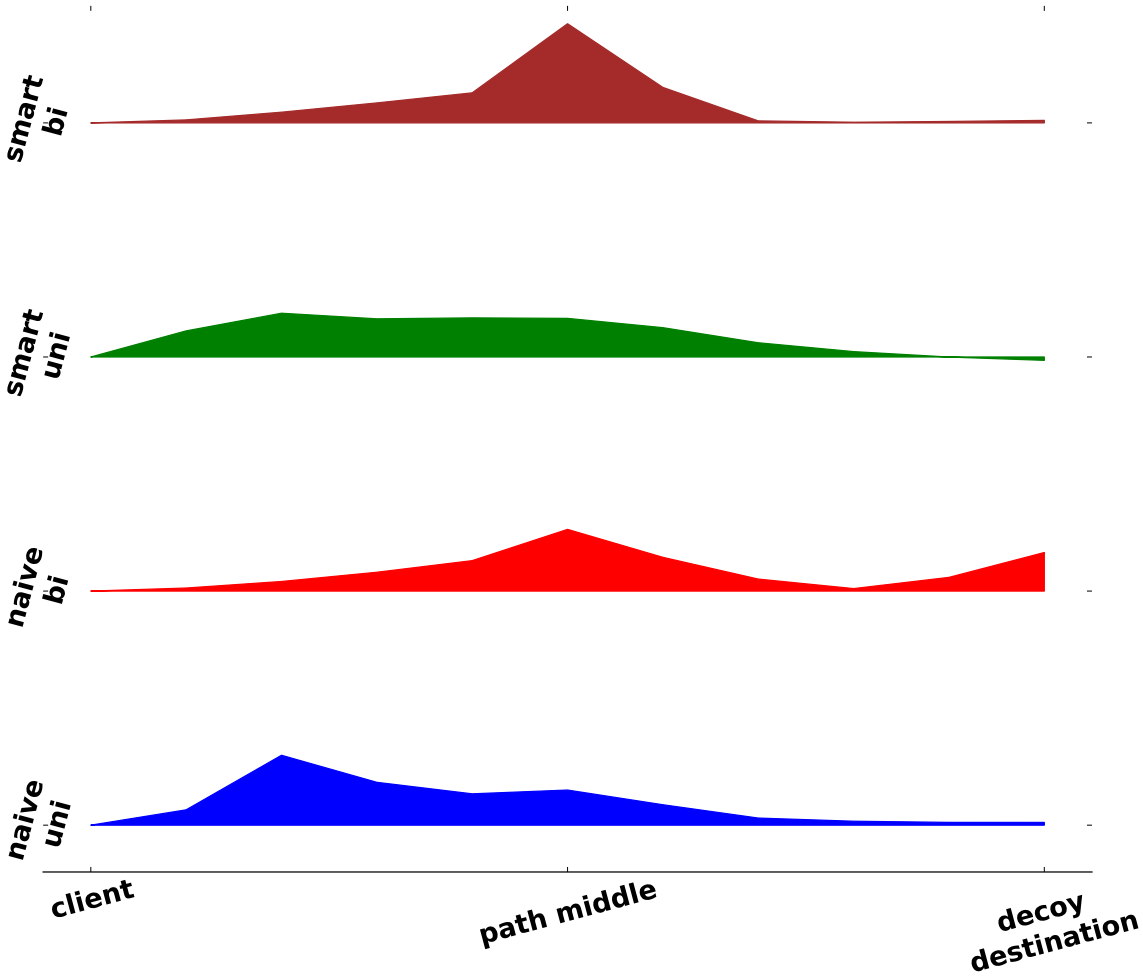

Figure 11: Analysis of the location of candidate DRs in the client-decoy destination paths. On the X-axis we identify three significant locations in each path: the client, the decoy destination and the middle of the path. Each curve in the graph represents the probability distribution function of the solutions with respect to different scenarios.

path properties (unidirectional, bidirectional) and the variable for filtering level (naïve,smart). For

each combination, we show the probability distribution function of the locations designed in the solution. The distribution is the median distribution across all countries.

We analyze in detail each distribution, starting from the bottom of the figure. In the naïve-unidirectional (naïve-uni) case the distribution leans more towards the client: we believe it is because the clients set presents a number of 'smaller' countries. For these smaller countries the best locations are closer to their borders. In the naïve-bidirectional (naïve-bi) case the majority of DRs are located in the middle of the path or near the decoy destinations. As we have discussed in § 5.4 forward and return paths may not present any intersections apart from the decoy destination. These results show indeed that many intersections only happen at the decoy destination but there is also a significant number of intersections deeper in the network. The smart-unidirectional (smart-uni) case presents a similar shape to the naïve-unidirectional one. The difference is a distribution more skewed towards the middle of the paths. The last distribution combines smart filtering and bidirectional paths. If we intersect the distribution from these previous two cases, the great majority of locations for DRs is deeper in the network.

In conclusion, it is evident that across different scenarios the locations predominantly chosen are in the middle of paths, deeper in the network. We associate such locations with large ASes, Tier-1 backbone ASes or Tier-2 regional ASes.

6 Related Work

The problem of DR placement in the network is similar to the well known and deeply studied problem of *monitor* placement.

Monitor placement is categorized into active and passive monitor placement.

Active monitor placement focuses on the deployment of devices capable of probing the network to infer its topological properties. In this space, Jamin *et al.* [19] propose the IDMaps project to calculate relative distances between hosts in the network. Even though their scope is different from ours, they discuss diminishing returns as the number of monitors increases. We have found a very similar result with DRs, and highlighted a 'break point' after which there is very small value in adding more decoy routers.

Horton *et al.* [16] focus on the problem of deploying active beacons to monitor the connectivity in the network. Their goal is to deploy the minimum number of beacons possible to infer the

status of every edge in the network. This problem is equivalent to the *min-k* problem we define in § 3.2. To solve it, they use a greedy heuristic based on the max connectivity of a node to other nodes in the network which they call *arity*. We draw similarities to Horton's work for their intent to correlate the beacons problem to the structure of the real network. In fact, Horton *et al.* claim that aiming for nodes with higher connectivity is an effective strategy to optimizing beacon placement. Similarly, we claim that DR deployment is most valuable in bigger ASes deeper in the network.

The focus of passive monitor placement is to maximize traffic monitoring and sampling while minimizing deployment of monitors. This problem is closer to the DRP problem. In fact, Suh *et al.* [23] focus on two monitor placement problems similar to $FPC - k$ §2.1 and $min - k$ §3.2: Budget Constrained Maximum Coverage and Minimum Deployment Cost. In the former, the goal is to maximize the number of flows sampled with a fixed amount of monitors. The latter is the dual problem: minimizing the number of monitors deployed to sample a given number of flows. Flows can be compared to paths in the decoy routing model. We differ from Suh's model as we focus on pairs and clients rather than flows. Furthermore, Suh *et al.* evaluate their model on smaller, synthetically generated networks while we rely on inferred data that spans across the entire network.

Jackson *et al.* [18] assess the problem of complete network monitorage. The goal of their analysis it to monitor every valid path between elements of a given universe. Similarly to our work, Jackson *et al.* approach the monitor placement problem by studying the AS-level network graph. They rank AS popularity by topological information of the network (out-degree of ASes involved). Then they propose to follow two strategies: depth first - instrumenting the N most popular links in the network - and breadth first - instrumenting the most popular inter-AS link of the top N ASes. Their results show that a breadth first deployment is more successful. We focus on instrumenting ASes rather than inter-AS links because we have found that for specific client sets, covering an AS completely yields higher coverage values.

7 Conclusion

Our analysis has shown many facets of the DRP problem. We have taken into consideration several client and decoy destination sets, unidirectional and bidirectional paths as well as a variation of

the original problem where filtering entities react to decoy routing.

The results highlight great similarities across solutions to different DRP problem scenarios. In particular, there always exists a small set of 'popular' locations where it is extremely valuable to have decoy routers. Equipping these locations accounts for the vast majority of paths between sets of clients and decoy destinations, no matter their size or geo-location. A look at the position of these location has shown that with great probability a DR deployment has to happen 'deep' in the network, in large Tier-1 and Tier-2 ASes. Outside of this set, adding more decoy routers yields diminishing returns. In other words, full coverage between clients and decoy destinations is very inefficient, if not infeasible.

The goal of decoy routing, though, is not to achieve total coverage of network, in large regional or national ASespaths between clients and network, in large regional or national ASesdecoy destinations, but to be pervasive enough to make it impossible for a filtering entity to effectively block its users. We have shown that this is achievable even in the case that every possible destination is considered a decoy, even if the DR has to lie on both the forward path and return path from the client to the decoy destination, and even if the filtering entity proactively tries to avoid decoy routing.

Overall, we believe that decoy routing is a valid response to network filtering and that its biggest challenge, the necessity to lie on the path between a client and a destination, can be solved efficiently.

References

[1] Global pass. `http://gpass1.com/gpass/`.

[2] Opennet initiative (ONI). `http://opennet.net/research/profiles`.

[3] RouteViews Project. `http://www.routeviews.org/`.

[4] CAIDA. The Cooperative Association for Internet Data Analysis. `http://www.caida.org`, 2010.

[5] CAIDA. AS Information Dataset. `http://as-rank.caida.org/data/2011.01/as2info.2010.08.txt`, 2010.

[6] CAIDA. AS Relationships Dataset. `http://www.caida.org/data/active/as-relationships/`, 2010.

[7] BBC News. Tehran blocks access to Facebook. `http://news.bbc.co.uk/2/hi/8065578.stm`.

[8] R. Deibert. Psiphon. `http://psiphon.civisec.org/`.

[9] X. Dimitropoulos, D. Krioukov, M. Fomenkov, B. Huffaker, Y. Hyun, k. claffy, and G. Riley. AS relationships: Inference and validation. *ACM SIGCOMM Computer Communication Review (CCR)*, 2007.

[10] R. Dingledine, N. Mathewson, and P. Syverson. Tor: The second-generation onion router. In *Proc. Usenix Security Symposium*, 2004.

[11] U. Feige. A threshold of ln n for approximating set cover. *ACM*, 1998.

[12] U. Feige, D. Peleg, and G. Kortsarz. The dense k-subgraph problem. *Algorithmica*, 2001.

[13] L. Gao and J. Rexford. Stable Internet routing without global coordination. *IEEE/ACM Trans. Netw.*

[14] S. Goldberg, M. Schapira, P. Hummon, and J. Rexford. How secure are secure interdomain routing protocols. In *"Proc. ACM SIGCOMM"*, 2010.

[15] D. Goldstein and M. Langberg. The dense k subgraph problem. *CoRR*, 2009.

[16] J. D. Horton and A. López-Ortiz. On the number of distributed measurement points for network tomography. In *Proc. ACM SIGCOMM*, IMC '03, 2003.

[17] A. Houmansadr, G. T. Nguyen, M. Caesar, and N. Borisov. Cirripede: Circumvention infrastructure using router redirection with plausible deniability. In *Proc. ACM Conference on Computer and Communications Security*, 2011.

[18] A. Jackson, W. Milliken, C. Santivanez, M. Condell, and W. Strayer. A topological analysis of monitor placement. In *Network Computing and Applications, 2007. Sixth IEEE International Symposium on*, 2007.

[19] S. Jamin, C. Jin, Y. Jin, D. Raz, Y. Shavitt, and L. Zhang. On the placement of internet instrumentation. In *IEEE INFOCOM*, 2000.

[20] J. Karlin, D. Ellard, A. Jackson, C. Jones, G. Lauer, D. Mankins, and W. Strayer. Decoy routing: Toward unblockable Internet communication. In *Proc. USENIX Workshop on Free and Open Communications on the Internet*, 2011.

[21] G. Nemhauser, L. Wolsey, and M. Fisher. An analysis of approximations for maximizing submodular set functions–i. *Mathematical Programming*, 1978.

[22] Y. Rekhter and T. Li. A Border Gateway Protocol 4 (BGP-4). RFC 4271 (Proposed Standard), 2006. URL `http://www.ietf.org/rfc/rfc4271.txt`.

[23] K. Suh, Y. Guo, J. Kurose, and D. Towsley. Locating network monitors: complexity, heuristics, and coverage. In *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, 2005.

[24] TOR, The Onion Router. Tor partially blocked in China. Blog post at `https://blog.torproject.org/blog/tor-partially-blocked-china`.

[25] E. Wustrow, S. Wolchok, I. Goldberg, and J. A. Halderman. Telex: Anticensorship in the network infrastructure. In *Proc. USENIX Security Symposium*, August 2011.