Quantitative Analysis of Incorrectly-Configured Bogon-Filter Detection

Jon Arnold,
Defence Science and Technology Organisation Adelaide
Jon.Arnold@dsto.defence.gov.au

Olaf Maennel,
Technische Universität Berlin
Deutsche Telekom Labs
olaf@maennel.net

Ashley Flavel, Jeremy McMahon, Matt Roughan
School of Mathematical Sciences
University of Adelaide
ashley.flavel@adelaide.edu.au
jeremy.mcmahon@adelaide.edu.au
matthew.roughan@adelaide.edu.au

***Abstract*—Newly announced IP addresses (from previously unused IP blocks) are often unreachable. It is common for network operators to filter out address space which is known to be unallocated ("bogon" addresses). However, as allocated address space changes over time, these bogons might become legitimately announced prefixes. Unfortunately, some ISPs still do not configure their bogon filters via lists published by the Regional Internet Registries (RIRs). Instead, they choose to manually configure filters. Therefore it would be desirable to test whether filters block legitimate address space before it is allocated to ISPs and/or end users. Previous work has presented a methodology that aims at detecting such wrongly configured filters, so that ISPs can be contacted and asked to update their filters. This paper extends the methodology by providing a more formal algorithm for finding such filters, and the paper quantitatively assesses the performance of this methodology.**

I. Introduction

It is common for Internet Service Providers (ISPs) to use so called bogon filters to eliminate traffic that is impossible (or bogus). A common example of such traffic is traffic originating from unallocated address space. Such traffic may arise as part of a (source address spoofing) DoS, Worm, or other attack [1], or as a result of address hijacking (as used by spammers [2], [3]) so such filters make a great deal of sense. However, address space allocations changes over time, as new address space becomes allocated and announced [4]. Bogon filters need to be kept up to date, but recent measurements [5] show that they are not. In a significant number of cases, new address space will be unreachable because of incorrectly configured filters.

Previous work [5] presented a methodology for detecting incorrectly configured filters. However, the approach has not been tested against complete ground truth data, nor has its quality with respect to the scale of measurements been measured. Furthermore, the approach relies on a rather ad hoc heuristic, and it is very likely this can be improved. In this paper, we propose an optimization approach to detecting incorrectly configured bogon filters. In order to solve the optimization problem we implement a genetic algorithm (GA), specifically that of [6]. We test this formulation using simulated BGP data and demonstrate its effectiveness in locating filters within the simulated BGP network.

Our findings suggest that the underlying approach is sound, given enough measurements, and a sparse set of incorrect filters. Without enough measurements, we cannot see the whole Internet, and so cannot infer the locations of all filters. If too many locations have filters, then there are unavoidable ambiguities. However, for many reasonable scenarios, we should be able to detect and localise a substantial proportion of incorrect filters, with a low false positive rate.

II. Methodology

The Internet is composed of a large number of independently administered networks (Autonomous Systems or ASs), coupled by the Border Gateway Protocol (BGP) into a single globe spanning entity. BGP [7]–[9] provides the glue that brings the Internet's diverse ASs together. Each AS is physically connected to other ASs at one or more locations. BGP is used across these links to exchange information about reachable IP prefixes. Routes are distributed between routers internal to an AS using iBGP. However, the distribution of routes, and the choice of "best" route is influenced by policies, a simple example of which involves filtering selected routing announcements to remove them from consideration. We primarily consider detecting this type of filtering here (though bogon filters may take other forms). Typical policies dictate the relationship between connected ASs, which commonly falls into one of the following two broad categories:

1) *Customer-Provider:* One AS (the customer) financially compensates the other AS (the provider) for connectivity to the remainder of the Internet.
2) *Peer-Peer:* A mutually beneficial relationship between two ASs to provide connectivity to each others' customers. No remuneration is required for traffic exchanged between the two peer ASs. An AS generally does not transit traffic between two of its peers, and BGP policies are used to enforce this condition, for instance, an AS would not advertise a route learnt from one peer to another peer.

Although these categories do not apply to all relationships, or BGP policies, they are highly illustrative. A peer-peer policy would be implemented, e.g., by not passing route information

from providers, to peers. This simple filtering prevents one peer from using the other for transit.

A. Detecting Incorrect Filters

The methodology for detecting incorrect filters proposed in [5] involves the following. First, the portion of address space that is intended to be allocated in the near future is temporarily assigned to the testing service. A set of *test-boxes* that are strategically scattered throughout the Internet announce *test-prefixes*. In addition each test-box announces an *anchor-prefix*. The *anchor-prefix* is a well established prefix, part of an address block that has been used for some time and is known to be reachable [10]. As the test-prefix and the anchor-prefix are announced from the same router, the paths through the Internet should typically be the same for both prefixes. Each of the test- and anchor-prefixes have a pingable IP address on a computer belonging to the testing service, called *test-IP* and *anchor-IP* respectively.

Traceroutes are then run from various locations against the test-IP as well as the anchor-IP. We call this probing technique *in-probes*. By comparing the two paths we can derive candidates that might potentially filter the test-prefix. There are a number of cases that might result from a pair of traceroutes from traceroute source to the target and corresponding anchor. We exclude pathological cases such as where the anchor is unreachable (indicating some type of broader routing pathology). Below we list illustrative cases along with some commentary on the inference available from each.

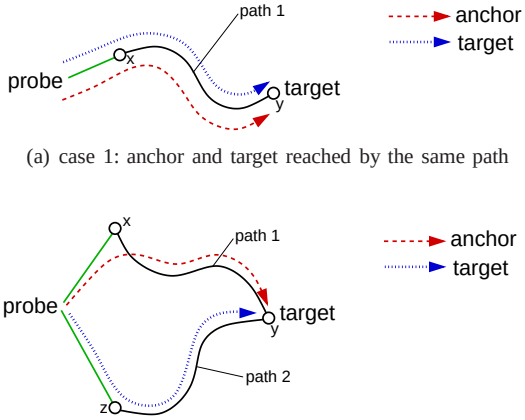
1) **anchor and target reached by the same path:** see Figure 1(a). In this case there is no filtering along the path.
2) **anchor and target reached by a different path:** see Figure 1(b). In this case some type of differential policy applies between test and anchor prefixes. The likely cause is a filter somewhere on the path of the test-prefix, though there are other possibilities:
   - traffic engineering [11].
   - BGP allows some non-locality in routing that could cause route changes due to filtering off of the main path.
3) **target not reachable:** In some cases filters may prevent an AS from even learning of a prefix, and in this case it will have no route to the destination at all. In this case we can clearly infer that a filter must exist somewhere on the test-prefix route.

We will use the simplest implications of each of these types of measurements to construct a set of constraints for the filters. The constraints list arise from

1) ASs where filtering cannot occur,
2) paths along which filtering is likely.

Our aim is to find a set of filters which match this possible set of constraints. Note that due to the possible sources of errors listed above, this list of constraints may not be consistent in practice, so it is desirable to have an approach that is error tolerant.

More importantly, it is likely that the feasible region specified by the constraints allows multiple solutions. In fact, many



(a) case 1: anchor and target reached by the same path

(b) case 2: anchor and target reached by a different path

Fig. 1. Example outcomes of traceroute measurements. Dotted and dashed lines show traceroute paths to the target and anchor respectively. Solid lines denote "paths" through potentially multiple ASs (not all shown).

solutions may be possible. In mathematical parlance, the problem is underconstrained. Hence, we need to select one of the possible solutions from the feasible region. In other network inference (tomography) problems, the method for dealing with this issue is to use a *prior* model of the expected structure of the solution (for instance the gravity model in traffic matrix inference [12]). We then seek the feasible solution that most closely matches the prior. The statistical term for this type of process is *regularization*. Often, the resultant problem can then be formulated as an optimization problem.

In this problem, we will use a prior such that our solution has maximal sparseness. That is, we seek the solution that uses the smallest number of filters to explain the observations. In reality the solution may not be the sparsest possible, but there are reasons to seek a sparse solution. Foremost, the Internet works. It would not if problematic filters were endemic. The defacto status of the Internet as a working network suggests that problem filters are rare.

So our approach will be to seek a solution which satisfies the observational constraints while maximizing the sparseness (minimizing the number of filters) needed in the solution. We can write this mathematically as follows.

B. Formalization

We have a graph $G = (N, E)$ with nodes N and edges E. Within this graph, we have a set of nodes, $S \subseteq N$, representing in-probe source nodes from which we may initiate trace-routes. Also, we have a set of nodes, $D \subseteq N$, representing the test-boxes where we announce the test and anchor prefixes.

Consider $s_i \in S$ and $d_j \in D$. In all graphs we will be considering, a path found using a trace-route from s_i to d_j for the anchor prefix announced from the test-box should exist. Let $a_{ij} \subseteq N$ be the set of nodes representing this path from s_i to d_j. Similarly, let $t_{ij} \subseteq N$ be the set of nodes representing the corresponding path for the test prefix, where we note that this set may be empty if no path exists, that is, the target is not reachable from our in-probe source.

Define

$$f_n = \begin{cases} 1, & \text{if node } n \text{ may be filtering the test prefix,} \\ 0, & \text{otherwise.} \end{cases} \quad (1)$$

Using the above definition and the path node-sets defined for pairs (s_i, d_j) we may write down some constraints for the location of filters in our node-set N. All nodes that are in t_{ij} must not be implementing any filtering for the test prefix and so, for all $s_i \in S$ and $d_j \in D$,

$$f_n = 0 \qquad \forall n \in t_{ij}. \tag{2}$$

If the path node-sets differ, then those nodes on the path to the anchor prefix that are not on the path to the test-prefix are candidates for filtering the test prefix. As such, we have that

$$\sum_{n \in a_{ij}} f_n \geq 1, \qquad \text{when } a_{ij} \neq t_{ij} \tag{3}$$

for all $s_i \in S$ and $d_j \in D$. Equation (3) dictates that if the paths differ, there must be at least one filter on the path for the anchor prefix. Note that some of these nodes will be automatically excluded by equation (2) if present in t_{ij}.

Given a sparsity criteria for determining the location of filters, we attempt to solve the binary optimization problem

$$\text{minimize } F = \sum_{n \in N} f_n \tag{4}$$

subject to the definitions and constraints given in equations (1), (2) and (3). The objective function given in equation (4) is aimed at minimizing the number of nodes in the network that filter the test prefix(es), due to the practical assumption that filtering is relatively rare.

The above formalization is somewhat of a simplification of the real problem, although it can be extended to allow for more complex filtering rules and hence additional constraints. We aim, however, to learn if solving this type of optimization problem is a viable approach for locating filters. The primary goal of the investigation contained in this report is to analyze the accuracy of inferred filter locations while varying the frequency of in-probe sources and test-boxes throughout the network. Given a set of in-probe sources, S, and a set of test-boxes, D, we may solve this optimization problem for this particular set of measurements to find a minimal set of filtering nodes that can explain the observed paths. This simplified problem will provide the appropriate intuition without unnecessary complications. These complications, however, could be added to a real system without major difficulty.

C. Solution

We use the optimization problem defined above to find the filter locations. There are various approaches for solving this optimization problem, but it is an integer programming problem and NP-hard. Hence, any viable solution must be obtained using heuristics, and will therefore be an approximation. Note, however, that our optimization criteria (sparsity) is an approximation in the first place, and so finding the *optimal* solution is not as important as finding a good solution within a reasonable amount of time.

The approach we use to solving the optimization problem is the Genetic Algorithm (GA) outlined in Hadj-Alouane and Bean [6]. This approach has several advantages in this context. Apart from the standard features of a GA (flexibility with respect to optimization objective function, ease of programming, and monotonic convergence), the GA approach inherently generates a *population* of solutions. This population may have multiple different, but equally good (or close to as good) solutions. This population of good solutions can give us a clear idea of the potential for multiple different solutions for some measurements (multiple potential solutions are an inevitable result of partial measurements).

We begin by initializing the population with random solutions for the problem at hand. We then score and rank the newly generated population. To score a specific element of the population, we have two contributing components. The first is simply the number of filters in the solution. The second is the number of measurement constraints violated, multiplied by some penalty multiplier, λ, determined by our perceived requirement that all constraints of the optimization problem be satisfied. The score of the solution is thus the sum of these components, and we rank the entire population from the lowest (best) score to the highest. Note that a low score indicates a sparse solution that satisfies a large number of the optimization constraints.

Once a population has been scored and ranked, we check to see if we wish to continue with the generation process of the GA. If so, we generate a new population using 3 distinct techniques. The first is to clone the top p_c% of the population, which guarantees monotonic convergence of the GA to an eventual optimal solution. Then, p_r% of the new population is randomly generated, a process akin to migration and thus diversity of the newly generated population is maintained. The remaining solutions required for the new population are generated as offspring from the previous population. For this purpose, we have chosen a parameterized (biased) uniform crossover of parents from the previous generation, which generates two offspring, and then the offspring with the best score is added to the new population.

When this population generation is complete, the new population is scored and ranked and the process continues until some termination criteria is met. This criteria is often a fixed number of generations, although for our analysis we have implemented a common early termination criteria whereby if the best solution of some number of successive generations remains constant, then it is likely that the optimal solution has been found, and the process may be terminated.

In preliminary analysis, the GA performs well, providing solutions in seconds on problems where MATLAB's optimization toolbox hits its iteration limit and returns no solution.

III. Results

A. Test Methodology

Prior work on this problem has used real data obtained from measurement experiments on the Internet [5]. However, this does not (i) give us a complete set of ground truth data, (ii) the ability to perform multiple realizations in order to obtain statistics with which to measure the performance of our approach. So, our approach to testing the algorithm is to use simulation. Of course, simulating the Internet is non-trivial. We use C-BGP [13], [14], to perform simulations because it allows us to simulate interdomain routing at the granularity

needed for this problem. Policies are realized using BGP communities and filters (as they might be in a real network), and we can simulate routing decisions including diversity and connectivity within each AS, without the unnecessary details of individual protocol packets. This allows us to run much larger simulations than other tools.

We simulate a tiered model of the Internet. That is, ASs fall into tiers, with tier-1 providers representing the "top" level providers who provide transit services to the next lower-layer, and so on down through the tiers. Lower-level tiers represent stubs, i.e., networks that connect users to the Internet, but don't provide transit services to other ASs. The model is implemented through BGP policies that dictate that links between ASs fall into one of the previously defined categories: customer-provider and peer-peer with matching BGP policies. The real Internet has an approximate tiered structure which reflects these AS business relationships, but the exact structure of these relationships is still a research question. Apart from the top and bottom levels there is considerable argument about the best model for tiering, so the intermediate layers are not so clearly delineated. In our model we therefore use only three tiers: tier-1, tier-2 and tier-3/stub ASs. It is also important to realize that ASs are not single points – they consist of multiple routers, and may have complex interconnections, and that this *does* change the behaviour of the system [15].

In the real Internet there are more than 26,000 ASs, of which a few dozen might be considered tier 1, down to maybe 5000 stub ASs. However, we will not use topologies of the same size as the Internet so that the causality within the simulation is clear, and we can quickly simulate all routing tables. We use AS topologies that consist of "only" 150 ASs but 575 routers. These topologies are large enough to accommodate a tiered structure and complex interconnections between ASs. Altogether, we have 5 tier-1s, 20 tier-2s and 125 stub ASs. We interconnect the ASs in the following manner: Tier-1 ASs are connected by a full mesh of peer-peer links. Each tier-2 AS is connected to x tier-1 providers where x is a random number between 1 and 5 (the number of tier-1 ASs). For at least one of these AS-level links we assign a customer-provider policy, and we randomly choose the policy for the other links. As a result tier-2 ASs are connected with up to 10 other tier-2 ASs by peer-peer links. For stub ASs, we toss a coin to decide whether it is to be dual- or single-homed. If it is single-homed, we randomly choose a tier2-ASs to connect it via a customer-provider link. For multi-homed stubs we randomly choose two tier-2 ASs and configure one link as peer-peer and the other as either peer-peer or customer-provider.

ASs cannot be seen as atomic entities that consist of a single router [15]. Rather, multiple routers per AS are needed to allow for path diversity. Yet, not all ASs have equally complex internal topologies. Tier-1s often have more complex networks than tier-2s and tier-3s. Therefore, in our simulations only tier-1s and tier-2s have multiple routers: tier-1 have 30 routers and tier-2 ASs have 15 routers. All routers within an AS are connected by iBGP sessions. Since in reality peer-peer AS edges rarely consist of a single link between two ASs, we configure multiple peering links for each AS-level edge by choosing a random number between 2 and 5 which is upper bounded by the number of routers in the ASs incident to the edge. For each edge we then pick a random router within each incident AS.

To capture the diversity of routes we originate one prefix for each destination AS and then compute the BGP routing tables for each router. Paths are computed by taking the standard concatenation of a series of such routing tables providing a set of AS paths between the 150 ASs. These represent the paths for the anchor prefixes and we denote this AS path set, A, and it forms the base against which we compare other path sets where filtering is implemented. In the simulations, a bogon filter is assumed to be consistent across an AS so no (newly advertised) routes can use this AS. We denote a set of AS paths that have been constructed in this manner as T, indicating that it relates to test-prefixes.

B. Performance Results

We perform a set of simulations varying the number of filters (from 2-10), test-boxes (from $|D| = 1$-10 though results from 9-10 are omitted due to space constraints and because they continue the obvious trend), and in-probe sources (from $|S| = 5$-90). In each case we perform 100 simulations choosing the set of test boxes and in-probe sources randomly. We do not allow ASs with filters to be used as observations points (either test boxes or in-probe sources) because this might make the problem too easy, and to further simplify the problem we do not allow a test-box site to also be the location of an in-probe.

In each simulation we construct the network, and thence a set of measurements. We then use the GA to solve the optimization problem described earlier. We experimented with the parameters of the GA, and found that the following worked as well as any others: the population size was 100, and we terminate the algorithm if 40 successive generations have the same 'best' solution. In the population generation process we clone the top 10 solutions, randomly generate 1 solution at each new population generation stage, and the 'bias' parameter in the offspring generation is set to 0.8. Finally, the constraint penalty multiplier $\lambda = 3$. With these parameters, the average computation time for all runs performed in this section is just over half a second.

Results for 2 filters are shown in Tables I-II. The table shows that with few observations it can be hard to find all of the filters, but that as the number of observation points increases, we can find all of the filters in all but a few cases. The result is a direct reflection of the fact that if none of the paths we observe pass through a filtering AS, then we cannot know that this AS is filtering. However, the false detection rate shown in Table II is very low. We do not find filters where there are none. As the number of filters increases from 2-6 (false detection rate results omitted), we observe the same findings. The proportion of filters identified depends on the number of ASs observed (see Tables III and IV).

When we increase the number of filters to 8 (or more) we start to see new phenomena. Tables V-VI show the results. The false detection rate, while still small, is no longer negligible. This results from a small number of locations in the topology where the combination of filters results in an ambiguity. That

TABLE I
2 FILTERS: MEAN NUMBER OF CORRECTLY IDENTIFIED FILTERS.

$\lvert D\rvert \rightarrow$ $\lvert S\rvert \downarrow$	1	2	3	4	5	6	7	8
5	0.28	0.36	0.68	0.82	0.87	1.03	1.01	1.28
10	0.38	0.68	0.84	1.25	1.26	1.35	1.39	1.57
15	0.52	0.82	1.14	1.32	1.49	1.51	1.62	1.72
20	0.51	0.95	1.26	1.36	1.63	1.72	1.67	1.71
25	0.79	1.15	1.39	1.55	1.67	1.64	1.89	1.84
30	0.61	1.13	1.51	1.61	1.66	1.81	1.77	1.92
35	0.67	1.09	1.44	1.63	1.75	1.79	1.82	1.91
40	0.81	1.19	1.48	1.67	1.84	1.85	1.96	1.93
45	0.83	1.25	1.58	1.70	1.79	1.82	1.88	1.97
50	0.89	1.28	1.52	1.73	1.80	1.86	1.93	1.98
55	0.78	1.24	1.56	1.80	1.87	1.92	1.96	1.95
60	0.84	1.34	1.61	1.70	1.93	1.93	1.94	1.96
65	0.87	1.25	1.63	1.76	1.83	1.95	1.95	1.98
70	0.89	1.41	1.62	1.85	1.83	1.93	1.96	1.97
75	1.00	1.41	1.58	1.72	1.83	1.96	1.95	2.00
80	0.96	1.48	1.53	1.79	1.89	1.94	1.98	1.99
85	0.78	1.50	1.65	1.82	1.90	1.96	1.96	2.00
90	0.89	1.40	1.76	1.85	1.92	1.98	2.00	1.99

TABLE II
2 FILTERS: MEAN NUMBER OF INCORRECTLY IDENTIFIED FILTERS.

$\lvert D\rvert \rightarrow$ $\lvert S\rvert \downarrow$	1	2	3	4	5	6	7	8
5	0.20	0.03	0.01	0.00	0.02	0.03	0.03	0.03
10	0.18	0.04	0.00	0.00	0.07	0.02	0.00	0.00
15	0.12	0.00	0.01	0.00	0.03	0.01	0.01	0.00
20	0.14	0.01	0.00	0.02	0.00	0.00	0.01	0.00
25	0.16	0.06	0.00	0.00	0.01	0.01	0.00	0.00
30	0.23	0.00	0.01	0.00	0.00	0.00	0.00	0.00
35	0.13	0.03	0.00	0.00	0.00	0.00	0.00	0.00
40	0.22	0.01	0.00	0.01	0.00	0.00	0.00	0.00
45	0.24	0.02	0.00	0.00	0.00	0.00	0.00	0.00
50	0.16	0.01	0.01	0.00	0.00	0.00	0.00	0.00
55	0.12	0.05	0.00	0.00	0.00	0.00	0.00	0.00
60	0.08	0.04	0.00	0.00	0.00	0.00	0.00	0.00
65	0.14	0.01	0.00	0.00	0.00	0.00	0.00	0.00
70	0.13	0.00	0.00	0.00	0.00	0.00	0.00	0.00
75	0.17	0.00	0.00	0.00	0.00	0.00	0.00	0.00
80	0.12	0.00	0.00	0.00	0.00	0.00	0.00	0.00
85	0.08	0.02	0.01	0.00	0.00	0.00	0.00	0.00
90	0.09	0.01	0.00	0.00	0.00	0.00	0.00	0.00

TABLE III
4 FILTERS: MEAN NUMBER OF CORRECTLY IDENTIFIED FILTERS.

$\lvert D\rvert \rightarrow$ $\lvert S\rvert \downarrow$	1	2	3	4	5	6	7	8
5	0.43	0.63	0.95	1.11	1.11	1.30	1.38	1.61
10	0.61	1.01	1.09	1.64	1.63	1.83	1.96	2.17
15	0.86	1.28	1.72	1.98	2.10	2.01	2.35	2.38
20	0.80	1.41	1.83	2.07	2.36	2.39	2.40	2.56
25	1.26	1.76	2.06	2.23	2.50	2.50	2.61	2.71
30	1.17	2.04	2.41	2.40	2.65	2.78	2.73	2.88
35	1.44	1.95	2.34	2.57	2.81	2.81	2.96	3.09
40	1.54	2.15	2.45	2.65	3.00	3.02	3.10	3.25
45	1.61	2.38	2.80	2.94	2.98	3.20	3.29	3.33
50	1.73	2.38	2.75	3.01	3.08	3.30	3.39	3.40
55	1.68	2.54	2.95	3.24	3.29	3.42	3.44	3.56
60	1.86	2.85	3.03	3.25	3.46	3.44	3.58	3.53
65	1.92	2.58	3.13	3.34	3.47	3.61	3.69	3.61
70	2.08	2.93	3.30	3.51	3.61	3.56	3.69	3.71
75	2.21	2.97	3.26	3.53	3.57	3.76	3.73	3.82
80	2.14	3.13	3.27	3.62	3.68	3.73	3.82	3.88
85	2.08	3.17	3.40	3.64	3.79	3.85	3.87	3.92
90	2.21	3.14	3.69	3.82	3.83	3.92	3.94	3.99

TABLE IV
6 FILTERS: MEAN NUMBER OF CORRECTLY IDENTIFIED FILTERS.

$\lvert D\rvert \rightarrow$ $\lvert S\rvert \downarrow$	1	2	3	4	5	6	7	8
5	0.62	0.86	1.26	1.39	1.48	1.80	1.79	2.17
10	0.95	1.55	1.59	2.01	2.21	2.39	2.68	2.91
15	1.30	1.80	2.32	2.60	2.84	2.65	3.16	3.19
20	1.54	2.09	2.61	2.80	3.17	3.26	3.28	3.54
25	1.97	2.52	2.88	2.98	3.47	3.56	3.53	3.74
30	2.01	3.03	3.42	3.29	3.67	3.98	3.82	4.01
35	2.35	3.05	3.48	3.77	4.12	4.06	4.29	4.28
40	2.62	3.40	3.74	3.83	4.25	4.27	4.45	4.43
45	2.75	3.63	4.07	4.31	4.35	4.64	4.76	4.69
50	2.94	3.73	4.11	4.46	4.53	4.74	4.90	4.91
55	2.84	3.96	4.43	4.67	4.85	5.00	4.94	5.17
60	3.24	4.34	4.64	4.78	5.06	5.01	5.27	5.17
65	3.29	4.12	4.83	4.92	5.11	5.34	5.37	5.27
70	3.48	4.63	4.97	5.30	5.32	5.27	5.35	5.47
75	3.75	4.57	4.99	5.25	5.33	5.55	5.54	5.60
80	3.68	4.88	5.05	5.46	5.46	5.55	5.69	5.77
85	3.77	4.99	5.23	5.52	5.71	5.68	5.79	5.79
90	3.93	5.06	5.58	5.74	5.72	5.86	5.92	5.97

is, given the actual set of filters, no set of measurements could distinguish the actual set of filters from a least one other alternative. A simple example of such an ambiguity could be a set of ASs which occur in series (with no other connections). Assuming we do not have a monitor in these ASs, we can never tell which is causing the filtering because all could explain a route which avoids the series of ASs. The GA naturally provides a population of potential filtering solutions. We can use this population to examine whether there is some ambiguity in the potential solutions.

The other phenomena that we observe occurs when there are 10 or more filters. In this case we can sometimes explain the observations (even with a nearly complete set of measurements) using a smaller number of filters than actually exist. The example above of a series of ASs serves to illustrate this case as well. If filters exist in all of the series of ASs, these filters may be explained by a single filter in one AS, and so our approach, which searches for the sparsest solution to the problem (the solution with the least number of filters) will return only a subset of the total filters. Again the fact that the GA returns a population of solutions to the filter location problem can provide insight into the potential for multiple filters. However, the underlying assumption of our work is that incorrect filters are sparse. Our approach works when the number of filters is more than 5% of the total population of the network. If more than 5% of networks have incorrect filters, this translates to over 1300 filtering ASs. This is not a problem to be debugged, so much as an epidemic.

TABLE V
8 FILTERS: MEAN NUMBER OF CORRECTLY IDENTIFIED FILTERS.

$\lvert D\rvert \rightarrow$ $\lvert S\rvert \downarrow$	1	2	3	4	5	6	7	8
5	1.06	1.46	1.94	2.03	2.31	2.70	2.82	3.16
10	1.51	2.39	2.47	2.91	3.29	3.63	3.83	4.19
15	2.06	2.60	3.48	3.95	4.15	4.05	4.52	4.73
20	2.48	3.25	3.81	4.21	4.48	4.86	4.90	5.17
25	2.92	3.82	4.24	4.42	4.86	4.96	5.12	5.48
30	3.11	4.30	4.81	4.81	5.16	5.59	5.51	5.75
35	3.51	4.42	4.92	5.40	5.82	5.74	6.01	6.06
40	3.73	4.74	5.13	5.28	5.88	5.97	6.23	6.24
45	3.56	4.91	5.57	5.89	6.05	6.41	6.54	6.46
50	3.77	5.00	5.80	6.11	6.24	6.48	6.69	6.76
55	3.86	5.32	5.97	6.25	6.61	6.72	6.74	6.95
60	4.37	5.48	6.12	6.34	6.70	6.79	7.12	7.04
65	4.14	5.55	6.35	6.53	6.89	7.13	7.16	7.14
70	4.56	5.73	6.54	6.81	7.04	7.00	7.21	7.32
75	4.47	5.54	6.56	6.89	7.09	7.35	7.46	7.49
80	4.70	5.97	6.55	7.15	7.13	7.30	7.42	7.58
85	4.66	6.47	6.79	7.14	7.35	7.38	7.58	7.62
90	4.59	6.39	7.05	7.32	7.39	7.59	7.71	7.83

TABLE VI
8 FILTERS: MEAN NUMBER OF INCORRECTLY IDENTIFIED FILTERS.

$\lvert D\rvert \rightarrow$ $\lvert S\rvert \downarrow$	1	2	3	4	5	6	7	8
5	0.53	0.54	0.59	0.44	0.59	0.75	0.57	0.67
10	0.80	0.86	0.77	0.61	0.79	0.67	0.64	0.64
15	0.80	0.80	0.91	0.70	0.81	0.60	0.74	0.64
20	1.12	0.98	0.91	0.71	0.74	0.87	0.67	0.73
25	1.14	0.98	0.74	0.87	0.73	0.78	0.58	0.77
30	1.09	1.01	0.88	0.73	0.72	0.69	0.72	0.65
35	1.18	0.89	0.78	0.93	0.89	0.80	0.79	0.56
40	1.14	1.00	0.81	0.82	0.61	0.68	0.64	0.60
45	1.00	0.87	0.82	0.72	0.67	0.70	0.68	0.69
50	0.84	0.84	0.77	0.67	0.78	0.70	0.69	0.70
55	0.87	0.90	0.79	0.66	0.68	0.62	0.62	0.76
60	0.88	0.77	0.77	0.74	0.73	0.62	0.59	0.63
65	0.72	0.76	0.81	0.71	0.72	0.70	0.59	0.67
70	0.78	0.82	0.72	0.70	0.62	0.67	0.59	0.63
75	0.75	0.71	0.72	0.70	0.66	0.62	0.70	0.61
80	0.67	0.68	0.73	0.67	0.67	0.66	0.75	0.68
85	0.79	0.77	0.76	0.74	0.78	0.64	0.72	0.73
90	0.73	0.72	0.80	0.81	0.65	0.72	0.77	0.70

IV. CONCLUSION

This paper has made many simplifications – too many to list in the space available. We have aimed to examine the feasibility of the problem rather than the intricate detail required in a real solution. However, most of these details do not alter the fundamental problem, they simply increase the size of the optimization problem that needs to be solved. Thus we believe that the qualitative nature of our results will hold for more complex problems. Moreover, the GA has been chosen specifically because of the ease with which its objective function can be enhanced. One of the key features of GAs is their flexibility, and this will allow addition of new constraints and parameters as needed.

There are many issues we wish to investigate in the future. For instance, how the position of monitors in the inter-AS hierarchy impacts the results, and how features of the real routing system impact on these results. In addition, the current simulations concern only small network, and it will be interesting to consider how the GA approach scales with network size.

ACKNOWLEDGEMENTS

This work was funded in part by DSTO contract 4500639065. This work was performed while Olaf Maennel was at the University of Adelaide supported by ARC grant DP0557066.

REFERENCES

[1] Y. Chen, A. Bargteil, D. Bindel, R. Katz, and J. Kubiatowicz, "Quantifying network denial of service: A location service case study," in *ICICS '01: Proceedings of the Third International Conference on Information and Communications Security*, 2001.
[2] P. Boothe, J. Hiebert, and R. Bush, "How Prevalent is Prefix Hijacking on the Internet?," *NANOG 36*, February 2006.
[3] N. Feamster, J. Jung, and H. Balakrishnan, "An empirical study of bogon route advertisements," *ACM Comput. Commun. Rev.*, vol. 35, no. 1, pp. 63–70, 2005.
[4] G. Huston, "IPv4 Address Report," 2007. `http://www.potaroo.net/tools/ipv4/index.html`.
[5] R. Bush, J. Hiebert, O. Maennel, M. Roughan, and S. Uhlig, "Testing the reachability of (new) address space," in *INM'07: Proceedings of the 2007 SIGCOMM workshop on Internet network management*, (New York, NY, USA), pp. 236–241, ACM, 2007.
[6] A. B. Hadj-Alouane and J. C. Bean, "A genetic algorithm for the multiple-choice integer program," *Operations Research*, vol. 45, no. 1, pp. 92–101, 1997.
[7] Y. Rekhter and T. Li, "A Border Gateway Protocol." RFC 1771, March 1995.
[8] S. Halabi and D. McPherson, *Internet Routing Architectures*. Indianapolis, Indiana: Cisco Press, second ed., 2000.
[9] C. Huitema, *Routing in the Internet*. Prentice Hall, 2000.
[10] Z. M. Mao, R. Bush, T. G. Griffin, and M. Roughan, "BGP beacons," in *IMC '03: Proceedings of the 3rd ACM SIGCOMM conference on Internet measurement*, pp. 1–14, 2003.
[11] B. Augustin, T. Friedman, M. Curie, and R. Teixeira, "Measuring load-balanced paths in the Internet," in *ACM SIGCOMM Internet Measurement Conference*, (San Diego, CA, USA), October 2007.
[12] Y. Zhang, M. Roughan, C. Lund, and D. Donoho, "Estimating point-to-point and point-to-multipoint traffic matrices: An information-theoretic approach," *IEEE/ACM Transactions on Networking*, vol. 13, pp. 947–960, October 2005.
[13] B. Quoitin and S. Uhlig, "Modeling the routing of an autonomous system with C-BGP," *IEEE Network*, vol. 19, November 2005.
[14] "C-BGP." http://cbgp.info.ucl.ac.be/,.
[15] W. Mühlbauer, A. Feldmann, M. R. O. Maennel, and S. Uhlig, "Building an AS-topology model that captures route diversity," in *ACM SIGCOMM*, (Pisa, Italy), 2006.