Routing State Distance: A Path-based Metric for Network Analysis

Gonca Gürsun, Natali Ruchansky, Evimaria Terzi, and Mark Crovella
Department of Computer Science
Boston University

ABSTRACT

Characterizing the set of routes used between domains is an important and difficult problem. The size and complexity of the millions of BGP paths in use at any time can hide important phenomena and hinder attempts to understand the path selection behavior of ASes. In this paper we introduce a new approach to analysis of the interdomain routing system designed to shed light on collective routing policies. Our approach starts by defining a new metric for 'distance' between prefixes, which we call routing state distance (RSD). We show that RSD has a number of properties that make it attractive for use in visualizing and analyzing the state of the BGP system. Further, since RSD is a metric, it lends itself naturally to use in clustering prefixes or ASes. In fact, the properties of RSD allow us to define a natural clustering criterion, and we show that this criterion admits to a simple clustering algorithm with provable approximation guarantees. We then show that by clustering ASes using RSD, one can uncover macroscopic behavior in BGP that was previously hidden. For example, we show how to identify groups of ASes having similar routing policies with respect to certain destinations, which apparently reflects shared sensitivity to economic or performance considerations. These routing patterns represent a considerable generalization and extension of the notion of BGP atoms to the case where routing policies are only locally and approximately similar across a set of prefixes.

Categories and Subject Descriptors

C.2.3 [**Network Operations**]: Network monitoring; C.2.5 [**Local and Wide-Area Networks**]: Internet — BGP

Keywords

BGP, Interdomain Routing

1. INTRODUCTION

One of the principal challenges in working with measurements of Internet topology is to extract insight from massive, complex datasets. Many projects have collected extensive measurements of Internet topology at the router or autonomous-system (AS) level, but comparatively few tools have been developed to discover important patterns or structures in the resulting data. The lack of such tools makes it difficult to visualize the Internet topology, understand the routing behavior of ASes, and detect significant changes in Internet routing.

In this paper, we define a new metric for analyzing routing at the interdomain level, and we describe various kinds of useful data analysis that the metric enables. The metric is called *routing state distance (RSD)*; it measures 'closeness' between two prefixes in terms of *similarity of routing* to the prefixes.

The key idea behind *RSD* can be stated simply: the routing state distance between two prefixes is conceptually just *the number of ASes that disagree about the next hop* to the two prefixes. (Later we will make this notion precise and deal with the practical issues that arise in applying it to real BGP measurements.) The basic idea is illustrated in Figure 1. Figures 1(a), (b), and (c) each show the next hops from a set of ASes to different destinations. The routing state distance between the destinations in (a) and (b) is 3, because there are three ASes that choose different next hops between (a) and (b). Likewise, the routing state distance between the destinations in (b) and (c) is 5.

The key properties of *RSD* can be understood by contrasting it with typical measures of distance between ASes. The most common starting point is the AS graph – a graph in which nodes are ASes – and distance is traditionally thought of as some number of hops between ASes. In contrast, *RSD* is *not* defined in terms of a graph; rather, it is defined in terms of a set of *paths*. This has a number of advantages. First of all, it is well-known that measurements of the AS graph are highly incomplete (a recent review of this problem is [17]). By defining *RSD* in terms of AS paths rather than the AS graph, the problem of missing edges in the AS graph does not arise. This is a crucial advantage, given the large fraction of missing edges in AS graph measurements. Second, as we will show later, the set of paths to a prefix is a much richer characterization of the prefix's location than is its position in the AS graph. As a result, *RSD* can provide much more nuanced information about the relative location of two (or more) prefixes than can be provided by simply noting their position in a graph.

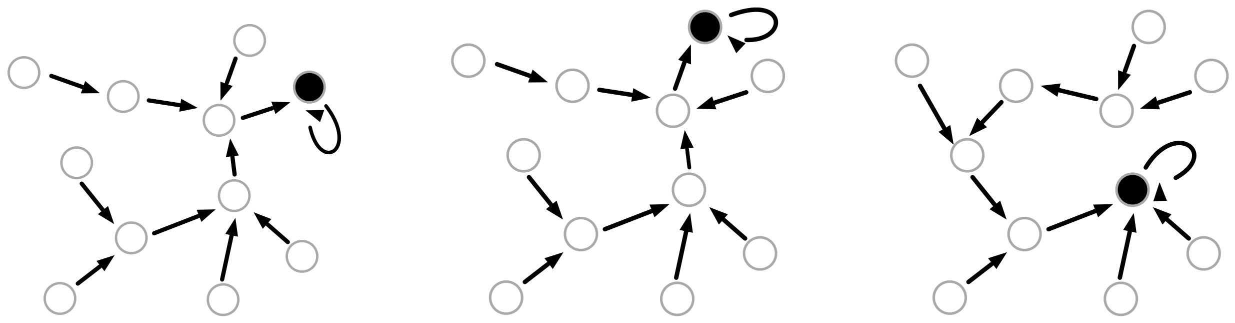


Figure 1: Example illustrating *RSD*. In each figure the filled node is the destination.

Having defined *RSD*, we next explore its utility. We show that it has a number of desirable properties. First, it is a metric (it obeys the triangle inequality), which makes it easy to reason about. Second, we show that it defines fine-grained neighborhoods — the number of prefixes that are within distance d of a given prefix declines smoothly with decreasing d. Third, we show that *RSD* generally exhibits *low effective dimension,* which means that 2-D visualizations can capture most of the information present in a set of *RSD* measurements. And fourth, we show that *RSD* has a natural clustering criterion, and that criterion has an associated clustering algorithm with provable approximation guarantees.

Most importantly, these four properties together allow us to develop an analytic toolkit that we use to uncover surprising patterns and behaviors that have not previously been documented in the interdomain routing system. Using the *RSD* toolkit we demonstrate the existence of collective routing behavior that we call *local atoms.* Roughly, a local atom is a set of prefixes that are routed similarly, but *only in some region* of the Internet. We show the existence of local atoms spanning a wide range of sizes, from macroscopic (Internet-wide) to microscopic. Although detecting these sorts of patterns in data is hard in general, we show that *RSD* and its associated clustering tools are natural and effective ways to uncover local atoms in BGP data.

Finally, we note that *RSD*, as a measure of distance between ASes, was first introduced by Gürsun et al. [10]. However, that paper used *RSD* for a specific purpose (as an input to a particular inference algorithm), and did not focus on the exploration of the theoretical and practical properties of *RSD* as a distance measure. In this paper, we fill this gap by studying the geometric properties of *RSD* and by defining notions of clustering appropriately tailored to this measure. Finally, we provide a thorough experimental study, which demonstrates the utility of *RSD* and the clusters identified by our clustering framework. Although our present study is motivated by the concept of *RSD* defined in [10], none of the results we present here were anticipated or discussed in that previous paper.

2. ROUTING STATE DISTANCE

In this section we define *RSD* starting from a general definition for arbitrary graphs. We then discuss how to customize *RSD* to address the practical issues that arise when applying it to BGP measurements.

2.1 Definition

To define RSD, we will assume a universe X of nodes, with $|X| = n$.[1] To fix a set of paths, we require that for each source-destination pair (x_1, x_2) there is a unique node x_3, which is the *next hop* on the path from x_1 to x_2. We denote this by $x_3 = \mathbf{N}(x_1, x_2)$. Note by following the nodes on $\mathbf{N}(\cdot, x_2)$ recursively, one will eventually reach x_2. We also assume that the next hop of every node x is the node itself; i.e., $\mathbf{N}(x, x) = x$. Thus, $\mathbf{N}$ encodes a set of paths that leads from each node to every other node, without any branches or loops.

We generally treat the function $\mathbf{N}$ as an $n \times n$ matrix. That is, we interpret $\mathbf{N}(x', x)$ as a matrix element that stores the next hop from node x' to node x. We also use $\mathbf{N}(x, :)$ (resp. $\mathbf{N}(:, x)$) to denote the x-th row (resp. column) of $\mathbf{N}$. We call $\mathbf{N}$ the *nexthop* matrix that encodes the paths over the set X.

Using the *nexthop* matrix, we can define the *Routing State Distance* (*RSD*) between two nodes x_1 and x_2 as follows:

$$RSD(x_1, x_2) = |\{x_i \mid \mathbf{N}(x_i, x_1) \neq \mathbf{N}(x_i, x_2)\}|. \quad (1)$$

That is, $RSD(x_1, x_2)$ is the number of positions where the columns $\mathbf{N}(:, x_1)$ and $\mathbf{N}(:, x_2)$ differ. Hence, by definition, *RSD* is an integer that takes values in $\{0, \ldots, n\}$. Referring back to Figure 1, each of the subfigures (a), (b), and (c) corresponds to a single column $\mathbf{N}(:, x)$ for different values of x. Intuitively, $\mathbf{N}(:, x)$ tells us what the 'direction' is to x from each node in X. Two nodes that appear to most other nodes to be in the same direction are considered close under *RSD*.

We now show that *RSD* satisfies the triangle inequality.

Proposition 1. *For any nodes x_1, x_2, and x_3,* $\mathrm{RSD}(x_1, x_2) \leq \mathrm{RSD}(x_1, x_3) + \mathrm{RSD}(x_2, x_3)$.

Proof. Assume the opposite, i.e., that $RSD(x_1, x_2) > RSD(x_1, x_3) + RSD(x_2, x_3)$. Then there must be a node x for which $\mathbf{N}(x, x_1) \neq \mathbf{N}(x, x_2)$, but for which $\mathbf{N}(x, x_1) = \mathbf{N}(x, x_3)$ and $\mathbf{N}(x, x_2) = \mathbf{N}(x, x_3)$, which is a contradiction. $\square$

Oftentimes, we use *rsd* to refer to the normalized value of *RSD*. That is, for every x_1, x_2 we define the normalized *RSD* as follows:

$$rsd(x_1, x_2) = \frac{1}{n} RSD(x_1, x_2). \quad (2)$$

[1]Note that the definition given here is essentially equivalent to that in [10], but has been recast in somewhat different terms for simplicity of discussion.

By definition, $rsd(x_1, x_2) \in [0, 1]$ and – given that it is a rescaling of *RSD* – it also satisfies the triangle inequality. Intuitively, the the *rsd* value between two nodes x_1 and x_2 encodes the *fraction* of positions in which the columns $\mathbf{N}(:, x_1)$ and $\mathbf{N}(:, x_2)$ differ with each other.

2.2 Applying RSD to BGP Analysis

Having defined *RSD*, we next seek to apply it to a dataset of publicly available BGP paths so as to compute the *RSD* between prefix pairs in the dataset. However, computing *RSD* from BGP data raises some implementation considerations.

The first issue concerns the distinction between ASes (which choose next hops, but are not themselves destinations) and prefixes (which are destinations, but do not make next-hop choices). In fact, our framework can adapt to this situation easily. We slightly redefine the matrix $\mathbf{N}$: while the columns of $\mathbf{N}$ correspond to prefixes, the rows of $\mathbf{N}$ correspond to ASes. Thus, to analyze BGP, $\mathbf{N}(a, p)$ needs to be defined as the next hop from AS a on the path to prefix p.

The next issue concerns the fact that publicly available BGP data consists (essentially) of paths from a set of monitor ASes to a large collection of prefixes. For any given AS-prefix pair (a, p), these paths may not contain information about $\mathbf{N}(a, p)$. We address this by *approximating RSD*. We make the following observation: some ASes have a much larger impact on *RSD* than others. For example, a stub AS a has a highly uniform row $\mathbf{N}(a, :)$. If the stub AS a has a small set of providers, then for many prefixes $p_1, p_2, ...$, $\mathbf{N}(a, p_1) = \mathbf{N}(a, p_2) = ...$ In the limit of a stub AS that makes use of only a single provider, the row $\mathbf{N}(a, :)$ will be constant. Note that a constant row in $\mathbf{N}$ has no effect on the *RSD* value of any prefix pairs. Since the majority of ASes are stub ASes, *most ASes contribute little information to RSD.*

Thus, we observe that we can approximate *RSD* using just a subset of all ASes. In particular we should use the ASes with many neighbors since these ASes have many next hop choices and therefore can contribute the most information to *RSD*. We call these ASes the *basis set.* We select the basis set by identifying those ASes with the largest number of neighbors in our data.[2] Luckily, such ASes tend to appear on many paths in the publicly available data; hence we can often find $\mathbf{N}(a, p)$ when a is in the basis set.

To address the case when $\mathbf{N}(a, p)$ is not available (for AS a in the basis set), we performed extensive studies of the effect of missing *nexthop* information on *RSD*. We found that proportional approximation yields results that work well in practice. This approximation allows the matrix $\mathbf{N}$ to include 'don't know' elements. We then approximate *RSD* by the fraction of *known* positions in which $\mathbf{N}(:, p_1)$ and $\mathbf{N}(:, p_2)$ differ, times the number of ASes in the basis set. This ensures that the approximation to *RSD* always ranges between zero and the size of the basis set.

One consequence of using proportional approximation to *RSD* is that it can introduce minor violations of the triangle inequality. However we find that in practice, such violations are quite small and do not impair our ability to reason about *RSD* or use it as if it were a metric.

The last issue is that for some AS-prefix pairs there is more than one next hop. This happens when an AS uses detailed routing policies (such as hot-potato routing) that are not strictly per-neighbor. That is, traffic destined for the same prefix may take different next hops depending on where it enters the network. We address this problem the same way as in [14], i.e., by introducing the notion of 'quasi-routers.' Using the algorithm in [14] we divide each AS in the basis set into a minimal set of quasi-routers such that for each (quasi-router, prefix) pair there is a unique next hop AS. Thus, the rows of $\mathbf{N}$ correspond either to ASes or quasi-routers of ASes.

[2]Two nodes are neighbors if they appear consecutively on any path.

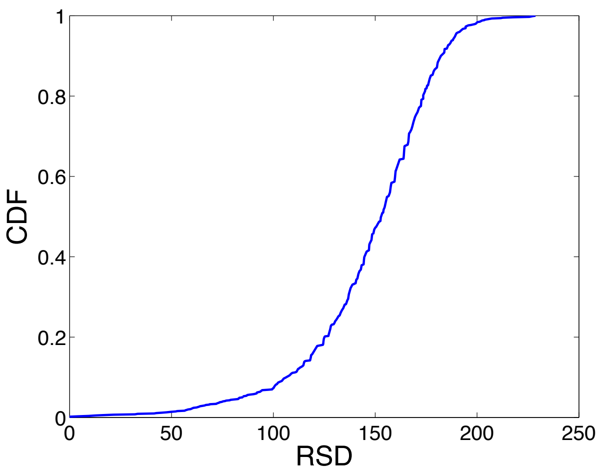


Figure 2: Distribution of $\mathbf{D}$ for 1000 prefix pairs.

Having addressed these issues, we can compute our version of *RSD* specialized for BGP, which we store in the $n \times n$ matrix $\mathbf{D}$. Clearly, the values of the cells of $\mathbf{D}$ depend on the number of rows in $\mathbf{N}$, i.e., the total number of ASes (or quasi-routers of ASes). For the rest of the discussion we will assume that we have m such ASes and therefore $\mathbf{N}$ is of size $m \times n$ and $\mathbf{D}(x, x') \in \{0, \ldots, m\}$. We also use $\widetilde{\mathbf{D}}$ to denote the normalized *rsd* for BGP.

3. DATASETS

To evaluate our methods, we use a collection of BGP tables collected on midnight UTC on December 6, 2011 from the Routeviews [18] and RIPE [16] repositories.

The full dataset is collected from 359 unique monitors; it consists of over 48 million AS paths, and contains 454,804 unique destination prefixes. However, the dataset does not contain paths from every monitors to every prefix. Rather, there is a subset of prefixes that appear in most tables, i.e., to which most monitors have paths. We chose a subset of 135,369 prefixes on this basis; these prefixes are typically present in most BGP tables in our dataset.

Next, we need to select a subset of monitors to serve as the basis set, as described in Section 2.2. We select the basis set by identifying 77 ASes with the largest number of neighbors in our data. Finally, we expand the basis set with quasi-routers to handle the cases where there is not a unique next hop to certain prefixes. This expands the size of the basis set from 77 to 243. Hence, our final $\mathbf{N}$ matrix is of size 243 $\times$ 135,369.

From $\mathbf{N}$ we can compute $\mathbf{D}$, our *RSD* metric based on BGP paths applied to prefixes. To illustrate the distribution

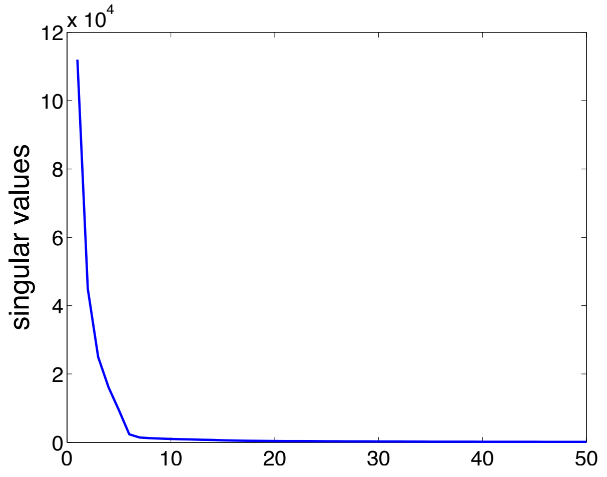


Figure 3: Singular values of $\mathbf{D}$.

of $\mathbf{D}$ values, Figure 2 shows the CDF of $\mathbf{D}$ for a randomly chosen set of 1000 prefix pairs. The figure shows three important aspects of *RSD* applied to prefixes. First, it takes on values in the range 0 to 243, because there are 243 ASes and quasi-routers in the basis set. Second, it varies smoothly – there are no sudden jumps or steps in the distance function. This is in contrast to a metric like hop distance, in which going from hop distance 1 to hop distance 2 encompasses a huge increase in the number of prefix pairs. Finally, the gradual slope on the left of the figure shows that *RSD* can make fine distinctions between prefix pairs. The number of prefixes in a neighborhood grows very slowly for small to moderate values of *RSD*, which means that *RSD* can be used to identify fine-grained groups of prefixes. This capability will be important when we use *RSD* for clustering later in the paper.

4. VISUALIZATION WITH RSD

In this section, we motivate the use of *RSD* for visualization and we demonstrate that visualization using *RSD* can yield useful insights in the analysis of interdomain routing.

4.1 Why 2-D Visualization is Meaningful

As described in Section 2, the information provided by RSD can be organized into a distance matrix $\mathbf{D}$ in which $\mathbf{D}(i, j) = RSD(i, j)$. We seek a way to effectively visualize the information contained in $\mathbf{D}$.

To do so, we start by observing that in practice, we find that $\mathbf{D}$ has *low effective rank.* That is, although $\mathbf{D}$ is a $n \times n$ matrix it can be well approximated by a matrix of rank r with $r \ll n$. A simple way to assess this is through a *scree plot,* which is a plot of the singular values of $\mathbf{D}$ in decreasing order. The sum of the first k squared singular values is equal to the fraction of total variance in $\mathbf{D}$ that can be captured in a rank-k approximation. Thus, the scree plot of $\mathbf{D}$ gives a direct assessment of the effective rank of $\mathbf{D}$.

We start by choosing 3000 prefixes at random from the set of all 135,369 prefixes in our dataset. We then form the 3000×3000 matrix $\mathbf{D}$ consisting of the RSD values for all prefix pairs. The scree plot of $\mathbf{D}$ is shown in Figure 3.

The Figure shows that $\mathbf{D}$ has very low effective rank – almost all of the the variation in $\mathbf{D}$ is captured in five dimensions, and even a rank-2 approximation to $\mathbf{D}$ captures more than half of $\mathbf{D}$'s total variance. (Results for other random samples look very similar).

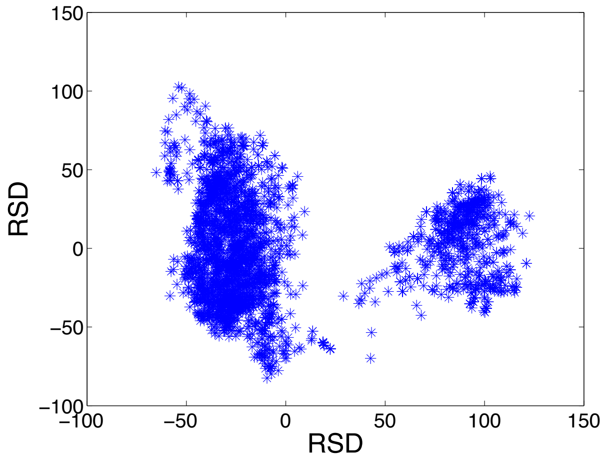


Figure 4: Visualization of *RSD* for 3000 randomly chosen prefixes. Axes are marked in units of *RSD* to give a sense of scale to the plots.

The fact that $\mathbf{D}$ has low effective rank is important for a number of reasons. First, it suggests that (as we will demonstrate) *RSD* captures specific phenomena – were the matrix $\mathbf{D}$ purely random, it would have high effective rank. Second, from a visualization standpoint, it indicates that a large fraction of the total information captured by *RSD* can be represented in a 2-D or 3-D visualization.

Thus, Figure 3 suggests that even a 2-D visualization of $\mathbf{D}$ should give a reasonably accurate picture of its information content. The usual way to construct such a visualization is by using *multidimensional scaling* (MDS) [19]. Given a set of items I and a set of item-item distances $\{d_{ij}\},\ i, j \in I$, MDS assigns each item a location $x_i,\ i \in I$ in some r-dimensional Euclidean space. The goal is that distances between items in Euclidean space should closely approximate the given set of input distances $\{d_{ij}\}$. That is, MDS seeks to minimize:

$$\min_{x_1,\ldots,x_{|I|}} \sum_{i<j} (||x_i - x_j|| - d_{ij})^2 .$$

When $r = 2$, the result can be plotted; in such a plot, distances (i.e., *RSD* values) are approximately preserved.

4.2 The Emergence of Clusters

We use MDS to visualize the randomly chosen set of 3000 prefixes. The results are shown in Figure 4; distances between points in this figure approximate *RSD*. The figure shows remarkable high-level structure: there are two large clusters, with the smaller cluster comprising about 23% of all prefixes. We find this same clustering quite consistently for *any* random sampling of prefixes; it seems to be an Internet-wide phenomenon.

The fact that Internet prefixes cluster into two distinct groups under the *RSD* metric is surprising. In fact, understanding the reason behind the presence of the two clusters in Figure 4 is important and motivates our subsequent analyses. Hence, we will now explore the cause of these two clusters in depth.

First, we consider exactly what causes clustering under

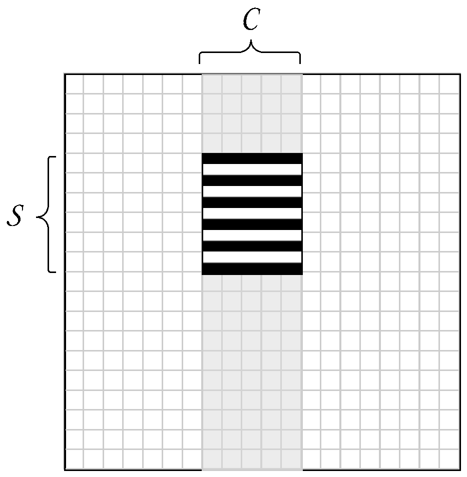


Figure 5: A cluster within the *nexthop* matrix $\mathbf{N}$.

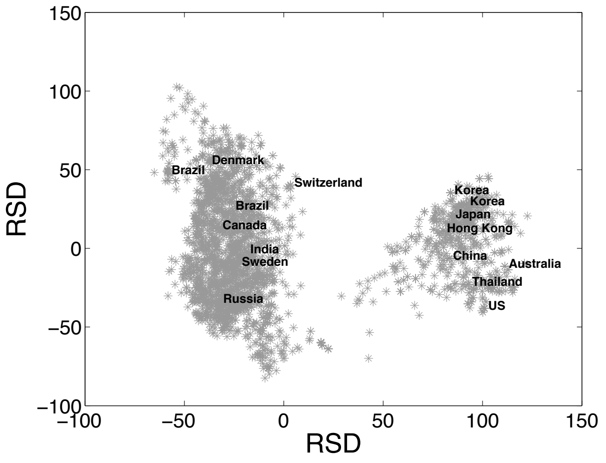


Figure 6: Locations of a random subset of prefixes.

RSD. Consider a cluster of prefixes C. This cluster corresponds to a set of columns of $\mathbf{N}$, i.e., $\mathbf{N}(:, C)$. Because the prefixes in C are close to each other in *RSD*, the columns of $\mathbf{N}(:, C)$ are similar to each other, at least in certain positions. This must be the case, by the definition of *RSD*.

This situation is shown in Figure 5. The figure shows the *nexthop* matrix $\mathbf{N}$, where cluster C is shown in gray. The fact that the columns are similar in certain positions S is signified by the horizontal bars. Note that for the columns to be similar, rows in S must be constant (or at least nearly constant).[3] The region where the rows are constant is the submatrix $\mathbf{N}(S, C)$. (We have assumed that the columns and rows of $\mathbf{N}$ have been reordered to bring out this structure.)

The key fact is that, in order for the prefixes in C to cluster together, there will typically be a subset of rows S with the following property: for any row within the submatrix $\mathbf{N}(S, C)$, the next hop AS in each cell must be (nearly) always the same AS. That is, the entries of $\mathbf{N}(S, C)$ are expected to be highly coherent.

Thus, we can identify a cluster with a coherent submatrix $\mathbf{N}(S, C)$. To understand what such coherent submatrices signify, we consider how they arise in the course of BGP routing. Expressed in terms of BGP, $\mathbf{N}(S, C)$ captures a set of source ASes (S) that *all make similar routing decisions* with respect to a set of destination ASes (C, i.e., the cluster). In Section 6, we use different measures to quantify and evaluate the coherence of the submatrix $\mathbf{N}(S, C)$.

We refer to the pair (S, C) as a *local atom.* A local atom is a generalization of the notion of BGP atoms [6]. Whereas a BGP atom is a set of prefixes that are routed the same way by all routers in the Internet, a local atom is a set of prefixes that are routed similarly *in some region* of the Internet (i.e., by the ASes in S). To be interesting, a local atom (S, C) should have a significant number of ASes in S and also prefixes in C.

To illustrate the concept of a local atom, we return to the example in Figure 4. The smaller cluster turns out to be the result of a local atom. We demonstrate this as follows. Sampling the prefixes in the smaller cluster, we find that they belong primarily to networks in the Far East, with a small portion belonging to networks in the US. For example, out of a random sample of 100 prefixes we find that 64% are Far East and Pacific Rim networks (including Korea, Japan, Taiwan, Thailand, Singapore, China, Hong Kong, Australia, and New Zealand) and 30% are from North America. However, there are also many Far Eastern (29%) and North American (20%) prefixes in the larger cluster (along with 33% European, which are almost completely absent in the smaller cluster).[4] Figure 6 shows examples of where a random selection of these prefixes occurs within the two clusters.

Given that Far Eastern and US prefixes occur in both clusters, why then should this specific set of Far Eastern and US prefixes group together in the smaller cluster?

The answer has to do with the next-hop decisions made by the source set S. There are 35 ASes in the S set; they are predominantly in Europe (52%) and North America (40%). These ASes prefer one particular provider for transit to the Far East and US. This provider is Hurricane Electric, ASN 6939 ('HE'). The overwhelming presence of AS 6939 as a next-hop results in the observed coherence of $\mathbf{N}(S, C)$. On the other hand, the set of ASes in S does not commonly agree on next hops to destinations in Europe. Therefore, these prefixes do not occur in the local atom. Instead, those prefixes appear in the larger cluster in Figure 4.

Figure 7 is a visualization of a portion of the *nexthop* matrix $\mathbf{N}$. In this plot, colors represent the five most popular next-hop ASes across both clusters. The prefixes (columns) consist of 50 samples chosen at random from each of the two clusters. The first 35 rows correspond to the ASes in set S and the remaining rows are other ASes shown for compar-

[3]Strictly speaking, it is possible to construct a cluster without a common set of similar rows; but such an arrangement is highly unlikely in practice as it requires a very specific structure. In all the cluster examples we study, we find that the prefixes cluster together because of a common set of similar rows.

[4]The assignment of a prefix to a geographic region was done by noting the AS that announced the prefix, and then combining information from the various routing registries along with inspection of network maps and peering relationships where available. In particular, for prefixes that were announced by internationally distributed ASes, care was taken to identify the region the prefix originated from.

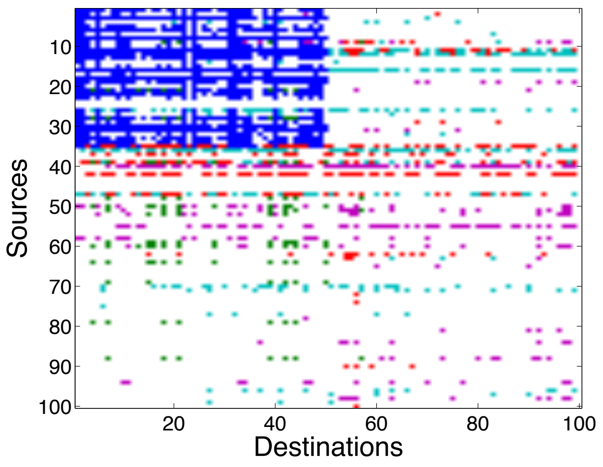


Figure 7: Coherence of the $\mathbf{N}(S, C)$ submatrix for the smaller cluster. The five most frequently-occurring ASes are shown in color: 6939 (Hurricane Electric, blue), 10026 (Pacnet, green), 3257 (TiNet, red), 3356 (Level3, turquoise), and 3549 (Global Crossing, magenta).

ison. The local atom (S, C) can be seen as the submatrix $\mathbf{N}(S, C)$ in the upper left part of the plot.

The figure shows the remarkable coherence of the $\mathbf{N}(S, C)$ submatrix. The similarity of the columns on the left side of the plot is the reason that those prefixes cluster together in Figure 4. It also shows that, while sources in S clearly distinguish between prefixes in the two clusters, other sources do not.

In fact, the prefixes that occur in the smaller cluster have the following property: if *any* path from a monitor to a prefix in our dataset passes through Hurricane Electric, then that prefix is in the smaller cluster. This remarkable fact is illustrated in Figure 8. The figure shows which prefixes can be reached from one or more monitors through three of the most commonly occurring ASes in our data: (a) Level 3, (b) Hurricane Electric, and (c) Sprint. The figure shows that what determines which prefixes go into the smaller cluster is whether the prefix can be reached through Hurricane Electric.

An example of routing to one such prefix is shown in Figure 9 (generated using BGPlay [4]). The prefix is one of those that falls into the smaller cluster. The figure shows how Hurricane Electric (ASN 6939) plays a special role with respect to this prefix for a large set of ASes. The presence of the smaller cluster is a result of routing patterns similar to Figure 9 for all the prefixes in the cluster.

To uncover the reasons behind this effect, we consulted with a number of the network operators whose BGP configurations contributed to the clustering (that is, operators of networks in the source set S). As a result, we can explain the reason for this unusual routing behavior with respect to Hurricane Electric: HE is a large ISP, but it has an open peering policy. That is, any ISP that has a presence in an exchange point in common with HE will be allowed to peer with HE at that exchange point [12]. HE is present in dozens of exchange points, mainly in the US and Europe (but with some in the Far East).

Note that most operators will prefer peer routes over provider routes in general. This implies that an ISP that peers with HE will typically use HE as the next-hop to reach any network that is a customer of HE (and not one of its own customers). Hence, we can identify S as largely consisting of networks that peer with HE, and D as the set of networks that are customers (or customers of customers, etc.) of HE.

Thus, the presence of two clusters among Internet prefixes is due to a large-scale phenomenon: for many ASes, Hurricane Electric is the preferred next hop for *any prefixes for which it provides transit.* Thus this local atom is a case of similar decision making by independent entities (ASes) when driven by common external factors; in this case, it seems that the particular, open peering policy used by HE is responsible for the observed similarity of routing behavior.

Hence, we conclude that clustering in *RSD* space has the potential to uncover local atoms, and local atoms are evidence of (generally unexpected) synchronization of routing decisions among certain ASes with respect to certain destinations. We then naturally ask the question: besides the macroscopic cluster shown in Figure 4, are there other local atoms, corresponding to other clusters, in our data? Presumably these clusters are at smaller scale and thus will be much harder to find. This motivates us to consider the problem of clustering in *RSD* space more analytically, and to look for efficient and effective methods of clustering for *RSD*.

5. CLUSTERING WITH RSD

Finding a natural clustering for prefixes using the *RSD* metric is challenging. Common clustering methods either operate over a continuous metric space (e.g., k-means) or others (e.g., k-median) require defining the notion of a 'representative' object for each cluster (i.e., the object that minimizes the sum of the distances to all other points in the cluster). Unfortunately, our data are not continuous; each data point is a column of the nexthop matrix and therefore its elements are categorical. Secondly, the notion of a representative in the *RSD* metric space is not clear. Further, most clustering algorithms require the number of clusters as input. Here we show that the *RSD* metric has a natural clustering formulation that does not have these drawbacks. Throughout this section, we will use the generic definition of *RSD* (or *rsd*) on a set of prefixes – which we refer to as nodes. In practice, we make the adaptations discussed in Section 2.2 and use $\mathbf{D}$ (resp. $\widetilde{\mathbf{D}}$) instead of *RSD* (resp. *rsd*).

5.1 The RS-Clustering problem

Given a set X of n prefixes, our goal is to produce a *partition* $\mathcal{P}$ of X; every prefix $x \in X$ belongs only to one cluster of $\mathcal{P}$, denoted by $\mathcal{P}(x)$. (Note that we will often overload notation and use partition names like $\mathcal{P}$ as labeling functions that map prefixes to clusters.)

Intuitively, a good partition satisfies the property that the routing state distance between two prefixes x and x' in the same cluster ($\mathcal{P}(x) = \mathcal{P}(x')$) should be minimized, while the routing state distance between prefixes x and x' in different clusters ($\mathcal{P}(x) \neq \mathcal{P}(x')$) should be maximized. This intuition can be captured in the following formal definition of the RS-Clustering problem.

Problem 1 (RS-Clustering). *Given a set of nodes* $X = \{x_1, \ldots, x_n\}$ *and the* $m \times n$ *nexthop matrix* $\mathbf{N}$*, find*

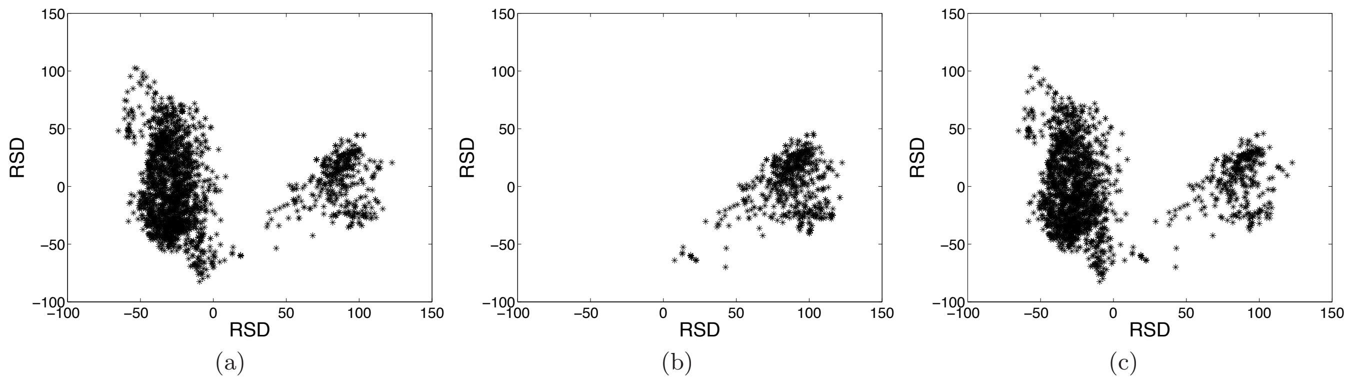


Figure 8: Prefixes that can be reached from one or more monitor ASes through (a) Level3, (b) Hurricane Electric, or (c) Sprint.

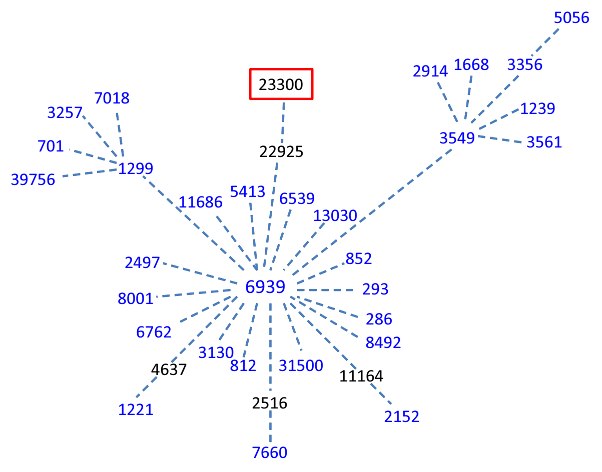


Figure 9: BGPlay snapshot of prefix 64.72.205.0/24 (origin AS: 23300) which occurs in the smaller cluster.

a partition $\mathcal{P}$ of the nodes in X such that

$$\text{P-Cost}(\mathcal{P}) = \sum_{\substack{x,x':\\ \mathcal{P}(x)=\mathcal{P}(x')}} \mathbf{D}(x,x') + \sum_{\substack{x,x':\\ \mathcal{P}(x)\neq\mathcal{P}(x')}} \left(m - \mathbf{D}(x,x')\right).$$

is minimized.

Observe that the definition of the RS-Clustering problem does not require the number of clusters to be part of the input. That is, RS-Clustering is *parameter-free.* This happens because the objective function of the clustering problem (i.e., the P-Cost function) is not guaranteed to decrease as the number of clusters increases. This is in contrast with the objective functions of many classical clustering problems (e.g., k-means, k-median etc). Hence, there exists an optimal number of clusters that minimizes the P-Cost function. A solution to RS-Clustering provides both the clusters of the prefixes as well as the optimal number of clusters as part of its output.

Despite the fact that the RS-Clustering problem is parameter free, its optimal solution cannot be computed in polynomial time. This is shown in the following proposition.

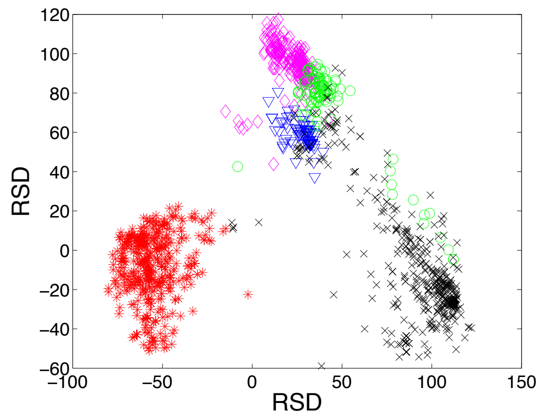


Figure 10: `Pivot` clustering of *RSD* with $\tau = 120$.

Proposition 2. *The* RS-Clustering *problem is NP-hard.*

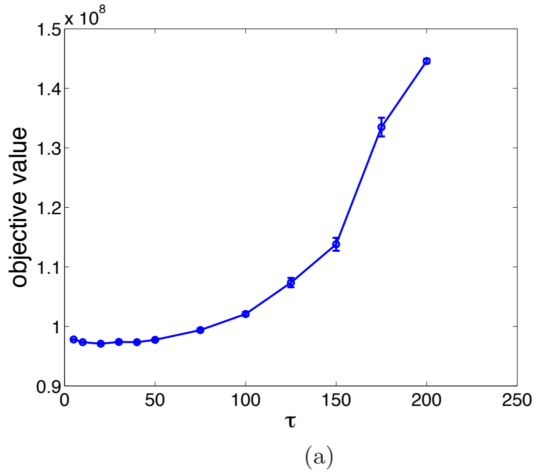
The proof of the above proposition is by a reduction from the Clustering Aggregation problem [2, 7, 8].

5.2 The `Pivot` algorithm

Based on the similarity between the RS-Clustering problem with the Correlation Clustering [3] and the Clustering Aggregation [2, 8] problems, we propose solving the problem using the `Pivot` algorithm, which was first proposed for solving the Clustering Aggregation problem.

The pseudocode of `Pivot` is shown in Algorithm 1. The algorithm takes as input the set of prefixes, their *RSD* values, and the value of a threshold parameter $\tau \in [0, m]$. The algorithm works as follows: starting from a random prefix x, it finds all prefixes that are within distance τ from x. All these prefixes are assigned in the same cluster – centered at prefix x. We call x the *pivot* of cluster C_x. The prefixes that are assigned in the cluster are removed from the set of prefixes X and the `Pivot` algorithm is reapplied to the remaining subset of prefixes that have not been assigned to any cluster.

Observe that `Pivot` requires the precomputation of all pairwise *RSD* distances. Given that these distances are known, the running time of `Pivot` is $O(n^2)$.



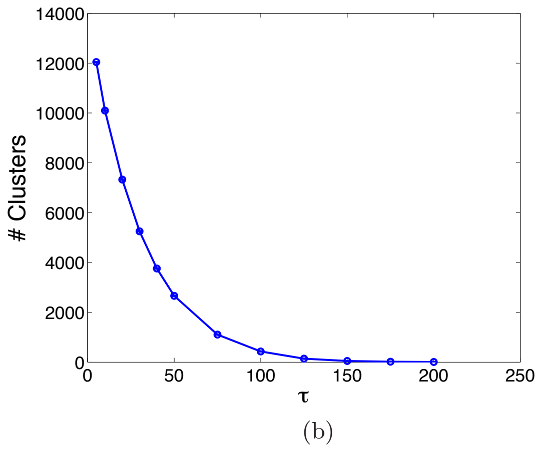


Figure 11: (a) Value of P-Cost and (b) number of clusters as clustering threshold τ varies.

Algorithm 1 The `Pivot` algorithm .

```
    A set of prefixes $X = \{x_1, \ldots, x_n\}$ and a threshold $\tau \in [0, m]$.
    A partition $\mathcal{P}$ of the prefixes
1: pick a random prefix $x \in X$
2: create a cluster $C_x = \{x' \mid \mathbf{D}(x, x') \le \tau\}$
3: $X = X \setminus C_x$
4: Pivot$(X, \tau)$
```

The quality of the solution output by `Pivot` can be measured using the P-Cost function. An interesting observation is that a small rewriting of the P-Cost function reveals that it is identical with the optimization function used for the Correlation Clustering problem [2, 3]. Hence, using the results of Ailon et al. [2] we can state the following:

Proposition 3. *For $\tau = m/2$, the* `Pivot` *algorithm is is an expected 3-approximation algorithm for the* RS-Clustering *problem.*

Observe that `Pivot` is a randomized algorithm since at every recursive call it picks a random prefix to play the role of a pivot.

6. APPLICATIONS

In this section, we illustrate how the solutions of RS-Clustering obtained using `Pivot`, automatically extract local atoms of $\mathbf{N}$.

We start by applying `Pivot` using the threshold $\tau = m/2$ as suggested by Proposition 3. This translates into $\tau = 120$ in our data.

Five large clusters identified by `Pivot` are shown in Figure 10. The sharpest separation is shown by the cluster in the lower left. Upon inspection, we find that the lower left cluster is in fact the Hurricane Electric cluster which was described in detail in Section 4. Note that whereas previously the Hurricane Electric cluster was identified manually, in this case it is extracted automatically through the use of `Pivot`. This provides good validation of the RS-Clustering problem definition and our proposed solution obtained via `Pivot`.

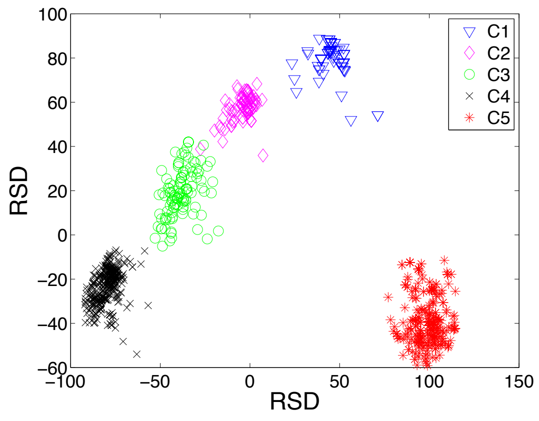


Figure 12: `Pivot` clustering of *RSD* with $\tau = 50$.

While Figure 10 shows that the Hurricane Electric cluster is clearly separated from the other prefixes, the other clusters are not so well separated. In fact, although `Pivot` with $\tau = m/2$ has a provable approximation bound, it is entirely possible that the algorithm may find a better solution with a different value of the threshold. We can assess the quality of a clustering simply by computing the value of the objective function P-Cost. Figure 11(a) shows how the objective function varies by decreasing the threshold below $\tau = 120$. It shows that at a threshold τ of about 50, the quality of the clustering levels off, and below 25 or so it starts to climb again. Furthermore, Figure 11(b) shows the number of clusters found at each threshold, and shows that below a τ value of 50 the number of clusters becomes very large. In fact, below $\tau = 50$, many clusters are just singletons which are not interesting as local atoms. Hence we next apply `Pivot` using a threshold value of 50.

Five of the largest clusters found with a threshold of 50 are shown in Figure 12. (Note that these clusters are not the same as those shown in Figure 10; the two figures show outputs of different clustering runs.) Compared to those in Figure 10, these clusters show much sharper separation, and we find that each of these corresponds to a local atom. To

Table 1: Statistics for the clusters in Figure 12.

	C_1			C_2			C_3			C_4			C_5		
Size of cluster (C)	150			170			126			484			375		
Size of source set (S)	16			9			7			8			15		
Destinations	Ukraine 83%			Romania 33%			India 93%			Russia 73%			US 74%		
	Czech Rep. 10%			Poland 33%			US 2%			Czech Rep. 10%			Australia 16%		
Dominant Next Hops	9002	21011	3549	5588	3549	1299	9498	3257	174	12389	3257	1273	6939	174	16735
Next Hop Density	0.26	0.23	0.11	0.33	0.16	0.11	0.30	0.10	0.06	0.31	0.07	0.06	0.54	0.06	0.03
Coherence	0.70	0.82	0.74	0.88	0.67	0.58	0.69	0.43	0.39	0.67	0.28	0.29	0.80	0.25	0.90

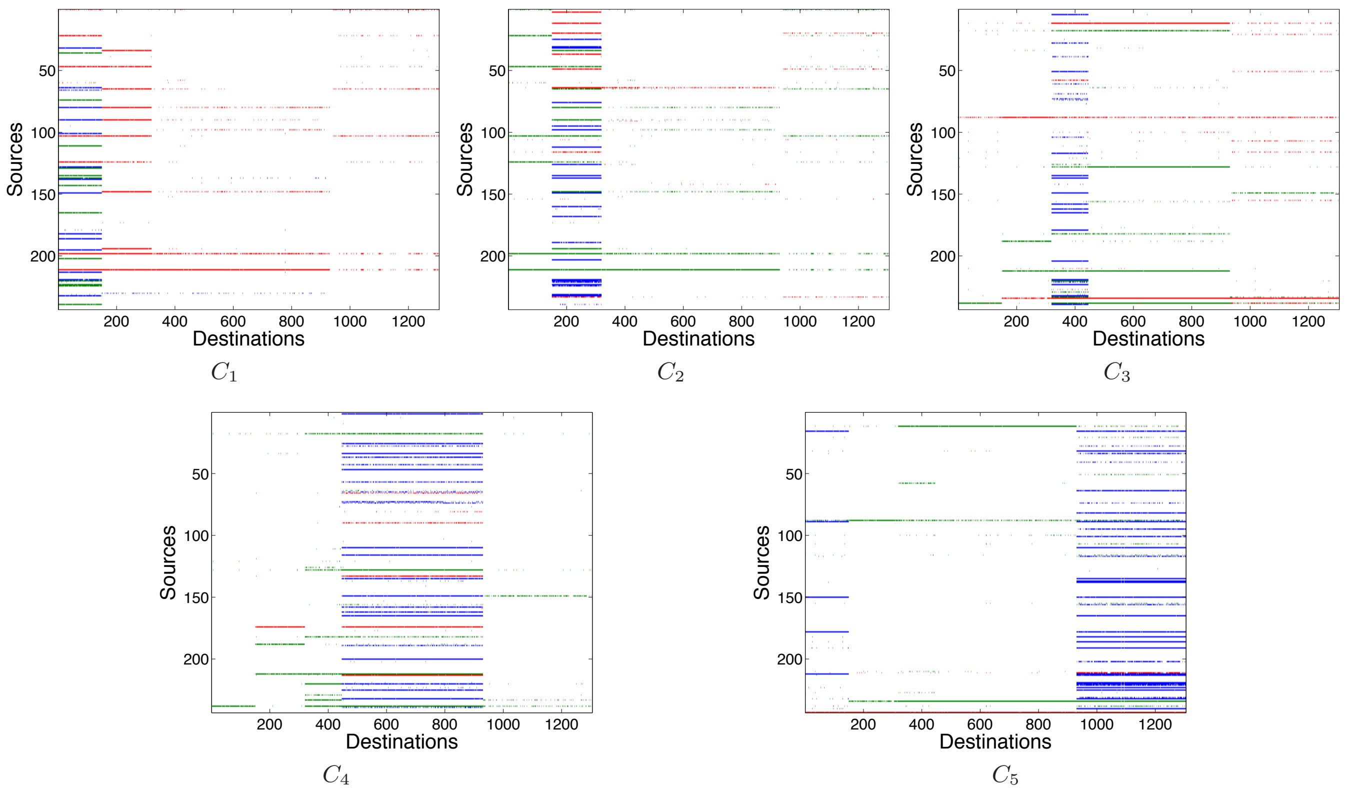


Figure 13: Visualization of the portion of **N** corresponding to the union of all five clusters. In each plot, we color the three most dominant next hops used in one cluster.

explore the nature of these local atoms, we start by characterizing them in Table 1.

The first two rows of the table give the size of the local atom. This is captured by the number of prefixes in the cluster (the size of C) and by the number of sources that show common routing behavior to those prefixes (the size of S). In each cluster, we find a significant number of ASes that show similar routing behavior to a large number of prefixes.

Next, we dive into the characteristics of each local atom. The next two rows characterize the geographic location of the prefixes (destinations). In each case, we have only listed the top-2 countries associated with the prefixes in the cluster, and we give the percent of prefixes that we find in each cluster from those countries. These rows show that in most cases, as much as 90% of the prefixes in a cluster are associated with only one or two countries. This shows that geography clearly influences the formation of local atoms.

The next three rows of Table 1 illustrate the nature of the 'common routing behavior' that the sources S exhibit with respect to the destinations C. The row labeled *Dominant Next Hops* lists the ASNs of the three most common next-hop ASes (in order of decreasing frequency) used by sources in S for destinations in C. The row *Next Hop Density* shows the fraction of entries in the cluster (across all 243 sources) that correspond to each of the three common next hops. And the row *Coherence* shows the fraction of entries *only in the submatrix* $\mathbf{N}(S, C)$ that correspond to each of the three common next hops. That is, the Coherence aims to quantify the cohesiveness of the nexthops appearing in the submatrix $\mathbf{N}(S, C)$.

The last two rows of Table 1 illustrate how common routing behavior is concentrated in the submatrix $\mathbf{N}(S, C)$. The much larger density of the dominant next hops used by sources in S as compared to among all sources shows that the sources in S are indeed making similar *nexthop* choices.

This is further illustrated graphically in Figure 13. In this figure, each plot is a view of the *nexthop* matrix $\mathbf{N}$. The same portion of $\mathbf{N}$ is shown in each plot, namely, the columns that correspond to prefixes forming the union of clusters C_1 through C_5. The difference between each plot is the choice of which next hop ASes are highlighted. In each plot, the three dominant next-hop ASes as given in Table 1 are colored in blue, green, and red, respectively. The figure shows how sharply the routing behavior of each local atom is defined. The common routing decisions made within each local atom appear as clearly isolated regions within the overall $\mathbf{N}$ matrix. This confirms our intuition of a local atom as equivalent to a coherent submatrix of $\mathbf{N}$.

7. OVERLAPPING CLUSTERING

The local atoms we have found so far have non-overlapping prefix sets. Consider the case of two coherent submatrices of $\mathbf{N}$ as shown in Figure 14. It is quite possible for BGP routing to result in this situation, but such local atoms can not be distinguished by the approach we have taken so far.

The `Pivot` algorithm outputs a *partition* of the prefixes. That is, every prefix x (column of $\mathbf{N}$) is only assigned to one cluster. However, as demonstrated by Figure 14, some columns should belong to both clusters, thus the expressiveness of partition-based clustering is limited. In this section, we propose a clustering that allows us to create clusters that can express such scenarios. Such clustering allows for *overlapping* clusters — the same node can be assigned to more than one cluster, and hence can capture the relationship shown in Figure 14.

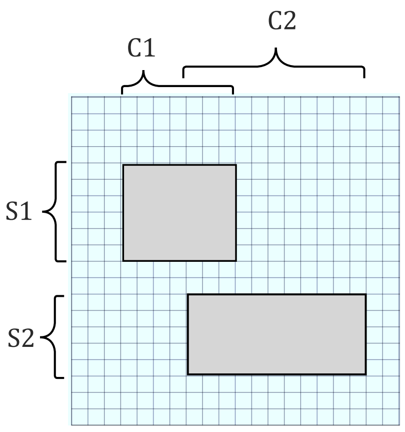


Figure 14: Overlapping clusters in $\mathbf{N}$.

Algorithm 2 The `Local` algorithm .

```
    A set of prefixes $X = \{x_1, \ldots, x_n\}$, the matrix of their
    normalized rsd distances $\widetilde{\mathbf{D}}$ and integer $p$.
    A labeling $\mathcal{L}$ that assigns at most $p$ labels (clusters) for
    every $x \in X$.
 1: $t = 0$
 2: for every $x \in X$ do
 3:     $\mathcal{L}_t(x) =$ Initialize($p$)
 4: end for
 5: while true do
 6:     $t = t + 1$
 7:     $\mathcal{L}_t = \mathcal{L}_{t-1}$
 8:     for every $x \in X$ do
 9:         $\widetilde{\mathcal{L}}_t(x) =$ NNLS($\widetilde{\mathbf{D}}, \mathcal{L}_t$)
10:         $\mathcal{L}_t(x) =$ Boolean($\hat{\mathcal{L}}_t(x), p$)
11:     end for
12:     if $|$L-COST$(\mathcal{L}_t) -$ L-COST$(\mathcal{L}_{t-1})| < \epsilon$ then
13:         return $\mathcal{L}_t$
14:     end if
15: end while
```

7.1 The OVERLAPPING-CLUSTERING problem

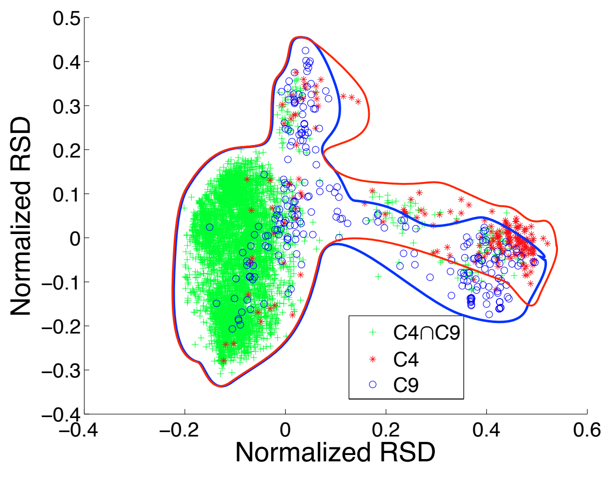
In this section, we define the OVERLAPPING-CLUSTERING problem. The output of this problem is a *labeling* $\mathcal{L}$, which assigns every node x to a *set* of clusters $\mathcal{L}(x)$. That is, every label corresponds to a cluster.

Since every node is assigned to a set (rather than to a single) cluster, we can compare a cluster-assignment for two nodes – e.g., x and x' – by comparing the corresponding labels. We do this by considering the *Jaccard distace* of the label sets $\mathcal{L}(x)$ and $\mathcal{L}(x')$, which is defined as follows:

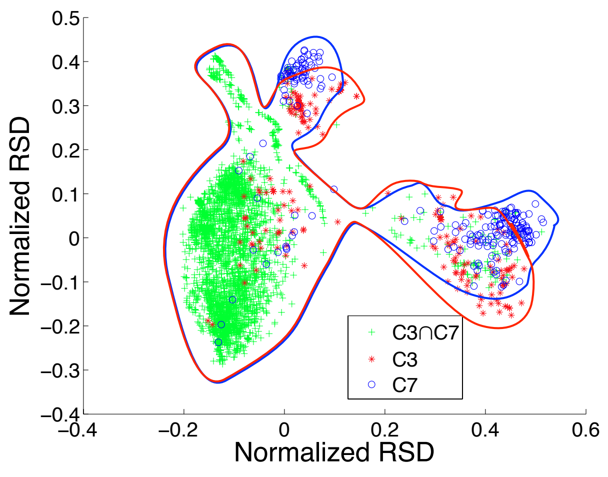
$$J\left(x, x' \mid \mathcal{L}\right) = 1 - \frac{|\mathcal{L}(x) \cap \mathcal{L}(x')|}{|\mathcal{L}(x) \cup \mathcal{L}(x')|}.$$

That is, the Jaccard distance captures the extent to which two label sets overlap, as a fraction of the total number of labels in both sets.

To capture the kind of relationship shown in Figure 14,



(a)



(b)

Figure 15: Cluster pairs (a) C_4/C_9 and (b) C_3/C_7. For each pair, the points in common are in green, while the points exclusive to each cluster are in red or blue. Cluster outlines are also overlaid for ease of interpretation.

the definition of the OVERLAPPING-CLUSTERING problem focuses on identifying a labeling of each prefix such that the Jaccard distance between the labels assigned to two prefixes approximates the *rsd* distance between the corresponding prefixes, to the maximum extent possible. Note that since the Jaccard distance is a real number in the interval $[0, 1]$ we seek to approximate the normalized version of *RSD*.

Our definition of OVERLAPPING-CLUSTERING builds on the recent work of Bonchi et al. [5]; its formal definition follows.

PROBLEM 2 (OVERLAPPING-CLUSTERING). *Given a set of nodes X, their $m \times n$* nexthop *matrix* $\mathbf{N}$ *and an integer p, identify a labeling of nodes $\mathcal{L}$ such that*

$$\text{L-COST}(\mathcal{L}) = \sum_{x,x' \in X \times X} \left| J\left(x, x' \mid \mathcal{L}\right) - \widetilde{\mathbf{D}}(x, x') \right|$$

is minimized and for every $x \in X$ it is guaranteed that $|\mathcal{L}(x)| \leq p$.

The definition of Problem 2 does not require the number of clusters in the output to be provided as part of the input. Rather, the problem requires as input the maximum number of clusters p in which a node can participate. Note that the L-COST function does not necessarily decrease as p increases. Thus, in theory there is an optimal value of p, which can be found by trying all possible values (i.e., $\{1, \ldots, n\}$) and report the clustering with the smallest L-COST value. In our experiments, we chose $p = 10$, because for $p > 10$ we observed no significant decreases in the objective function.

7.2 The `Local` algorithm

Bonchi et al. [5] showed that not only is Problem 2 NP-hard to solve, but it is also NP-hard to approximate. Therefore, we are content with efficient heuristics to minimize the L-COST objective function. In this section, we give a brief description of such a heuristic, which we call `Local`; this heuristic is an instantiation of the generic algorithm proposed in [5]. Although space does not permit a detailed description of our algorithm here, the code is available at: `CodeURL`.

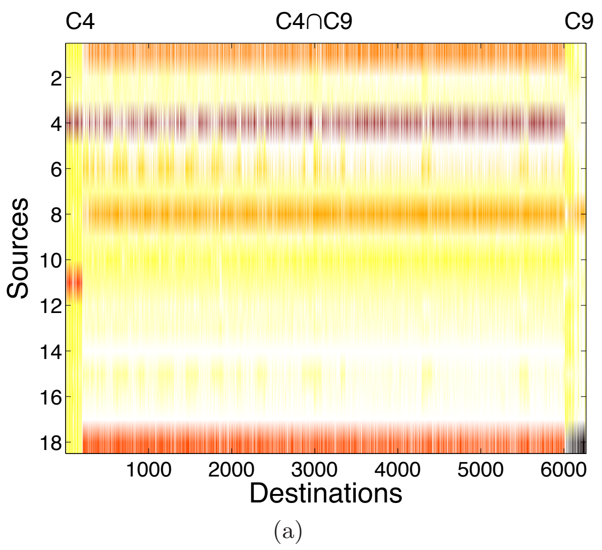
The pseudocode of the `Local` algorithm is shown in Algorithm 2. On a high level it uses a local-search heuristic, which starts with a random assignment of prefixes to clusters and iteratively improves it to reduce the value of the L-COST function. More specifically, at the first step (`Initialize` routine), every prefix is assigned a random set of at most p labels. Subsequently, the labels of each prefix x are appropriately changed so that the cost of the objective function improves. This is done by the `NNLS` and the `Boolean` routines. First, `NNLS` finds a real-valued assignment of labels to a node by solving a non-negative least squares problem. The output of this step is weighted association of the prefix with every cluster– denoted by $\widetilde{\mathcal{L}}$ in Algorithm 2. Then, the routine `Boolean` transforms the real-valued assignment into one-zero assigment. Thus, the output of this step is a set of labels (i.e., clusters) that are associated with prefix x. The details of these steps are described by Bonchi et al. [5]. The two steps are repeated, till a local minimum – with respect to the objective function L-COST – is reached.

7.3 Overlapping Clustering Results

To demonstrate overlapping clustering using *RSD* we apply `Local` to our data with $p = 10$. This allows any prefix to be a member of as many as 10 clusters. To understand the results, it is easiest to look at pairs of clusters. All pairs of clusters have some overlapping subset, so pairs of clusters illustrate the general power of overlapping clustering for our data.

From our 10 clusters, we focus on two pairs which we denote C_4/C_9 and C_3/C_7. We show the two pairs visualized using MDS in Figure 15.

The figure shows that, as intended, clusters overlap. In each case, a large number of prefixes are in both clusters. What is perhaps less expected is that points that are close to each other in *RSD* space as visualized in 2-D are not necessarily contained in the same cluster. For example, note the regions on the right side of each plot in which points belonging to distinct clusters are shown to be near in *RSD*.



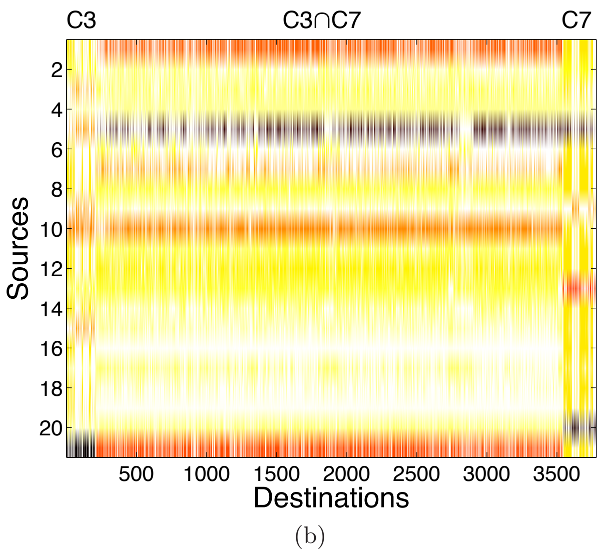


Figure 16: Submatrices of $\mathbf{N}$, with different nexthop ASes in different colors.

In fact, this effect is exactly what we had hoped overlapping clustering would capture. The reason for this effect is that point-pairs can be close to each other in *RSD* space *for different reasons.* Consider again Figure 14. There are two different sets of sources (S_1 and S_2) that have similar routing behavior and create coherent submatrices in $\mathbf{N}$. Each set of sources causes prefixes to cluster together, but independently so; the prefixes in C_1 are near each other in *RSD* space because of the similar routing decision of the ASes in S_1, while prefixes in C_2 are near each other in *RSD* space because of the similar routing decision of the ASes in S_2.

As a result, in contrast to the non-overlapping case, we do not find that clusters typically map to geographic sets or regions. Furthermore, clusters are not compact in *RSD* space as can be seen in Figure 15.

The overlapping clustering constructed by the `Local` algorithm allows us to discover the different sets of source ASes that have similar routing behavior. This is shown in Figure 16. The figure shows submatrices of $\mathbf{N}$ in which each next hop values has been given a color. These submatrices correspond to the cluster pairs C_4/C_9 and C_3/C_7. In each figure, we have ordered the columns so that the prefixes contained in each cluster are in the center, and the prefixes that are only in one cluster are on the sides. This allows us to visualize the sets of ASes that give rise to each cluster.

For example, in Figure 16(a), we can see that the AS in row 4 has the same next hop for all prefixes cluster C_4, but not for the prefixes that are only in C_9. Likewise, the AS in row 8 has the same next hop for prefixes in C_9, but not for those only in C_4. In the case of C_3/C_7, the cluster-defining rows are rows 5 and 10.

We conclude that these results give evidence that overlapping clustering (using an algorithm such as `Local`) can find clusters in which different sets of ASes each have cohesive routing behavior over overlapping sets of prefixes. This holds the promise of extracting a richer set of local atoms from BGP data.

8. DISCUSSION

Our sense is that *RSD*, owing to its simple definition, is a general tool that can be used in a variety of settings. These include further in-depth analysis of BGP, as well as its application to entirely different domains.

A promising direction for further application of *RSD* is to incorprate temporal analysis. Consider a set of nexthop matrices indexed in time, i.e., $\mathbf{N}_t, \mathbf{N}_{t+1}, \ldots$. In this setting, one can ask about the *RSD* of the same prefix at different points in time. For prefix x, this is simply the number of positions where $\mathbf{N}_t(:, x)$ and $\mathbf{N}_{t+1}(:, x)$ differ. Such a measure could have value over short timescales for detecting sudden, significant routing changes (such as a prefix hijacking) or over long timescales for characterizing the evolving routing structure of the Internet. Further, comparison of average *RSD* change across all or a selected set of prefixes from time t to time $t + 1$ could give a sense of overall rate of change and stability in the interdomain routing system.

Another direction for further analysis of BGP follows from the observation that the *RSD* matrix $\mathbf{D}$ has low effective rank. Previous work on traffic matrices has shown that this property naturally leads to an effective method for anomaly detection [13]. Our initial investigations indicate that this method can very effectively identify prefixes that have unusual connectivity patterns in BGP (such as CDNs).

Another consequence of the low effective rank property of $\mathbf{D}$ is that $\mathbf{D}$ may be amenable to matrix completion. That is, it may be possible to accurately *estimate* the *RSD* between two pairs of prefixes if enough measurements of *RSD* between other pairs is available. This could be useful in situations where only partial path information is available.

Going beyond the analysis of BGP, *RSD* can be applied in any situation in which paths between nodes are an available data source. For example, it could be applied to collections of traceroute measurements, information dissemination in social media, or infection propagation in social and communication networks.

9. RELATED WORK

It's been widely reported that publicly available BGP tables provide an incomplete view of the AS graph [17]. In this regard, it is important to develop tools that can analyze the network using only the BGP path data instead of the AS graph. The way we define RSD based on paths eliminates the problem of missing edges in the AS graph so that ambiguities due to missing links do not arise.

In addition, our paper touches on the problem of visualization of prefixes (or ASes) based on BGP data. The authors in [11] visualize the set of core ASes using placement based on AS degree and geographic location. Small scale visualizations of selected BGP paths are available through `BGPlay` [4] and `bgpviz` [15]. However, to our knowledge, no macroscopic visualization method relies directly on BGP paths as we do, despite the fact that paths are the fundamental unit of measurement in BGP. In addition, relatively little work has been done on clustering of prefixes or ASes; the work that has been done relies on the inferred AS graph rather than BGP paths [9].

Finally, a number of studies explore aggregation of prefixes. In [6], the authors show that there are many cases in which a collection of prefixes is routed the same way. Such collections, so called *policy atoms*, consist of prefixes that share the same BGP paths. In [1], the authors study methods for calculating policy atoms and their characteristics in detail. The main goal of these works is using policy atoms to reduce the sizes of BGP tables. Our work differs from these works in the sense that we do not require that the prefixes in the same *local atom* routed the same way. Instead we cluster the prefixes that are routed similarly *in some regions* of the Internet. In fact, we generalize and extend the notion of *atoms* so that it can be used to analyze the routing behaviors at different granularities.

10. CONCLUSIONS

In this paper we have developed a new set of tools for extracting insight from BGP measurements. We defined a new metric for measuring distance between prefixes, called *RSD*. This metric captures the notion of 'closeness' of two prefixes with respect to the global routing state of the Internet. As such, it is a natural tool for visualizing the relationship between prefixes. But going beyond visualization, we show that *RSD* can uncover surprising patterns in routing data. In particular, we show that *RSD* exposes situations in which large numbers of ASes make similar routing decisions with respect to specific sets of prefixes. We call this situation a local atom, generalizing the notion of a BGP atom. We show that local atoms exist at the macroscopic scale (one involves 23% of all prefixes in the Internet) as well as at much smaller scales. To expose local atoms, we develop and demonstrate the power of two new clustering methods specifically tailored for use with *RSD*. We conclude that the combination of (1) computing *RSD*, (2) visualizing *RSD* using MDS, and (3) clustering *RSD* using `Pivot` and `Local` together constitute a powerful toolbox for discovering patterns in massive datasets of BGP paths.

Code for the `Pivot` and `Local` algorithms, as well as code for computing *RSD*, is available at `http://csr.bu.edu/rsd`. That site also provides a list of *RSD* values for the prefix pairs used in this study.

11. ACKNOWLEDGEMENTS

This work was supported by NSF grants CNS-1017529, CNS-0905565, CNS-1018266, CNS- 1012910, CNS-1117039, by a GAANN Fellowship, and by grants from Microsoft, Yahoo!, and Google. The authors thank the IMC referees and shepherd for their help in improving the paper. The authors also appreciate helpful discussions with various network operators.

12. REFERENCES

[1] Y. Afek, O. Ben-Shalom, and A. Bremler-Barr. On the structure and application of bgp policy atoms. In *ACM SIGCOMM Workshop on Internet measurment*, 2002.

[2] N. Ailon, M. Charikar, and A. Newman. Aggregating inconsistent information: Ranking and clustering. *J. ACM*, 55(5), 2008.

[3] N. Bansal, A. Blum, and S. Chawla. Correlation clustering. *Machine Learning*, 56:89–113, 2004.

[4] Bgplay. `http://bgplay.routeviews.org`.

[5] F. Bonchi, A. Gionis, and A. Ukkonen. Overlapping correlation clustering. In *ICDM*, pages 51–60, 2011.

[6] A. Broido and k. claffy. Analysis of RouteViews BGP data: policy atoms. In *NRDM*, 2001.

[7] V. Filkov and S. Skiena. Integrating microarray data by consensus clustering. *Intern. J. on Artificial Intelligence Tools*, 13:863–880, 2004.

[8] A. Gionis, H. Mannila, and P. Tsaparas. Clustering aggregation. *TKDD*, 1(1), 2007.

[9] C. Gkantsidis, M. Mihail, and E. W. Zegura. Spectral analysis of internet topologies. In *INFOCOM*, 2003.

[10] G. Gürsun, N. Ruchansky, E. Terzi, and M. Crovella. Inferring visibility: Who's (not) talking to whom? In *SIGCOMM*, 2012.

[11] B. Huffaker and k. claffy. CAIDA's IPv4 and IPv6 AS Core AS-level Internet Graph. `http://www.caida.org/research/topology/as_core_network/`, 2010.

[12] Hurricane Electric. Hurricane electric peering policy. `http://he.net/peering.html`.

[13] A. Lakhina, M. Crovella, and C. Diot. Diagnosing network-wide traffic anomalies. In *SIGCOMM*, 2004.

[14] W. Mühlbauer, S. Uhlig, B. Fu, M. Meulle, and O. Maennel. In search for an appropriate granularity to model routing policies. In *SIGCOMM*, 2007.

[15] RIPE. bgpviz. `http://www.ris.ripe.net/bgpviz`.

[16] Ripe's routing information service raw data project. `http://www.ripe.net/data-tools/stats/ris/ris-raw-data`.

[17] M. Roughan, W. Willinger, O. Maennel, D. Perouli, and R. Bush. 10 lessons from 10 years of measuring and modeling the internet's autonomous systems. *IEEE Journal on Selected Areas in Communications*, 2011.

[18] University of oregon route views project. `http://www.routeviews.org`.

[19] W. S. Torgerson. Multidimensional scaling: I. theory and method. *Psychometrika*, 17:401–419, 1952.