An Efficient Approach to Multi-level Route Analytics

Ang Chen[§], Edmond W. W. Chan[†], Xiapu Luo[§‡], Waiting W. T. Fok[§], and Rocky K. C. Chang[§]
Department of Computing[§], Shenzhen Research Institute[‡] Noah's Ark Lab[†]
The Hong Kong Polytechnic University Huawei Technologies, China
{csachen|csxluo|cswtfok|csrchang}@comp.polyu.edu.hk edmond.chan@huawei.com

***Abstract*—Contrasting multi-level routes (e.g., IP, subnet, AS levels) is an analytical primitive underpinning many applications, such as route asymmetry and/or diversity measurement, route change characterization, efficient route-tracing design, and others. We are the first to identify that current approaches incur redundant node comparisons, because they treat each level independently. We propose a new approach called `rtd` that eliminates the redundancy, therefore improves the analysis efficiency, by integrating all levels recursively. Our extensive evaluations on simulated traces and real data from Ark, FastMapping, and iPlane datasets show that `rtd` eliminates 85% comparisons on average and doubles the analysis speed. Finally, we design a route clustering application using `rtd`, and demonstrate how it aids the monitoring of an ISP transition.**

I. Introduction

Traceroutes [27] are among the readiest data in today's network measurement research. For example, CAIDA's Ark project [26] probes all routed /24 prefixes to construct a wide-coverage topology. FastMapping [12] performs high-frequency tracing to study the dynamics of load-balanced paths. DTRACK [13] issues intensive traceroutes on unstable paths to detect more changes. iPlane [33], on the other hand, traces the routes originated from hundreds of vantage points to predict paths accurately. Moreover, researchers can customize datasets with a plethora of traceroute variants, such as Paris-traceroute [4] that exposes all load-balanced paths, TraceNet [47] that explores the subnet-level paths, AS-level traceroute [35] that returns a path's AS-level sequence, reverse traceroute [30] that infers the backward IP path, NANOG-traceroute [2], IPv6 traceroute [1], and others.

Analyzing traceroute data efficiently poses a great challenge because of their Internet-sized volume. CAIDA's Ark alone has collected more than 10 billion IPv4 traceroutes, and 500 million more continue to arrive in each probing round. Ark, however, only characterizes an incomplete portion of the current Internet, and tracing through the immense IPv6 space [6] will undoubtedly multiply the traceroute volume. Moreover, many applications typically require processing multiple units of route data. For example, studying dynamics of an IP-level route or topology requires contrasting *multiple rounds of routes*. Studying their dynamics on IP-/router-/PoP-/AS-level requires contrasting *multiple levels of routes* (or *multi-level routes*). Generally, an n-round or n-level analysis will increase the load of computation by n times.

It is therefore not surprising that route analytics incurs differing amount of delays before actual applications could even start. For example, a round of route change characterization precedes the final diagnosis of submarine cable faults [10], and a round of route asymmetry quantification precedes the correlational studies with delay asymmetry (e.g., [39], [37]). Understandably, such analyses should not take more time than necessary, because processing delays can be associated with economic loss (e.g., an extra delay of 0.4 seconds costs $188 million for Google [49]). Therefore, accelerated analytics, even on a small scale, means much. Moreover, some time-critical applications (e.g., adaptive probing systems) cannot work accurately unless their route analytics is executed in real time. For example, Beverly et al. [8] finds that efficient route-tracing designs require each probing round to adapt online to the results of the previous ones, and offline "train-then-probe" mechanisms are prone to expire and miss IP interfaces.

We aim to accelerate the operation of *contrasting multi-level routes*, which can be described as: given two routes with multiple levels of labels (e.g., IP-/subnet-/AS-level), identify their differences on each level. This operation is an analytical primitive underpinning many applications, and we name some of them here.

- **Route asymmetry measurement**: contrasting forward and reverse routes on router- and AS-level helps quantify their asymmetry (e.g., [39], [37], [22]). The differences between multi-level routes can be further correlated with performance asymmetry (e.g., [39], [37]).
- **Route change characterization**: contrasting snapshots of a route on IP-, /24-, and AS-level (e.g., [43], [31], [10]) helps characterize the scope of route changes. The differences between multi-level routes can further facilitate studies of cable repairing [10], measurement of available bandwidth [31] and delay variations [39].
- **Route diversity measurement**: contrasting redundant routes in multi-homed or overlay networks on IP- and AS-level helps quantify the route diversity [21].
- **Efficient route-tracing design**: contrasting IP routes when constructing an IP-level topology helps reduce probes [8]. It is not difficult to see that similar saving can be applied to AS-/PoP-level routes for AS-/PoP-level topology construction (e.g., [44]).

We identify that existing designs of this primitive incur

978-3-901882-50-0

many redundant node comparisons, therefore a large overhead, because they contrast routes on each level independently. Consider two nodes N_x and N_y in routes x and y. If an AS-level comparison already identifies that N_x and N_y belong to different ASes, comparing them further on IP-level would be redundant. Unfortunately, current approaches are oblivious to this concern and compare them again on IP-level anyway. As a result, the repetitive comparisons increase applications' computational load, lengthen their analysis cycles, and hinder the timeliness of adaptive applications.

In this paper, we propose a novel contrasting approach that eliminates redundant comparisons, therefore improves the efficiency of multi-level route analytics for the aforementioned applications. We make three contributions:

1. We are the first to identify the computational redundancy in current approaches to contrasting multi-level routes, and expose that the root cause lies in their independent use of levels. (§II-A)
2. We propose `rtd` that integrates all levels recursively to eliminate the redundancy. With `rtd`, an n-level analysis does not have to increase the computational load by n times. Under an ideal case, `rtd` performs a fixed order of comparisons no matter how many levels we use. We implement it in 1000 lines of C code which will be released with this paper. (§II-B)
3. We evaluate `rtd` on simulated traces and Ark, iPlane, FastMapping datasets, and show that on average 85% comparisons performed by current approaches are redundant. `rtd` reduces the number of comparisons to 15% and doubles the analysis speed. (§III)

Although high-performance analytics on "Big Data" can also be implemented by distributed computing platforms [15], novel software solutions [41], massively parallel processing (MPP) databases [19], [45], or harnessing the cloud [3], `rtd` caters for "Big Route Data" and offers an economical approach that requires no installation of additional hardware or software suites. Moreover, `rtd` can also be applied atop other approaches. For example, after decreasing analyses' computational load by `rtd`, one can proportionally enlist less VMs in the cloud while still respecting the original deadline.

We then design a route clustering application using `rtd` and apply it to monitor an ISP transition in §IV, review related work in §V, and conclude our paper in §VI.

II. Contrasting Multi-level Routes

We first define the problem of contrasting multi-level routes formally. Consider a route x that is an ordered sequence of nodes $X_1X_2\cdots X_{|x|}$. Each node $X_i, i \in [1..|x|]$, has n levels of labels, and we write its t-th level of label as $L_t(X_i)$. The levels form an ordered set $\mathcal{L} = \{L_1, \cdots, L_n\}$ with a transitive relation $\succ$ that reads as "contains", and $L_1 \succ L_2 \succ \cdots \succ L_n$. $L_t \succ L_{t+1}$ means that if $L_t(X_i) \neq L_t(X_j)$ then $L_{t+1}(X_i) \neq L_{t+1}(X_j)$, but $L_{t+1}(X_i) \neq L_{t+1}(X_j)$ does not imply $L_t(X_i) \neq L_t(X_j)$, $\forall i, j \in [1..|x|]$. Many existing works (e.g., [39], [37], [22], [43], [31], [10], [21]) employ topological levels $\mathcal{L} = \{$AS-level, subnet-level, IP-level$\}$ to study Internet topology or path dynamics, and assume that AS-level $\succ$ subnet-level $\succ$ IP-level. We adopt this setting and denote $AS(X_i)$/$SN(X_i)$/$IP(X_i)$ as the AS-/subnet-/IP-level label of X_i. However, our discussion extends similarly to levels chosen from other dimensions, such as organization-level $\succ$ AS-level [9], country-level $\succ$ city-level, and others.

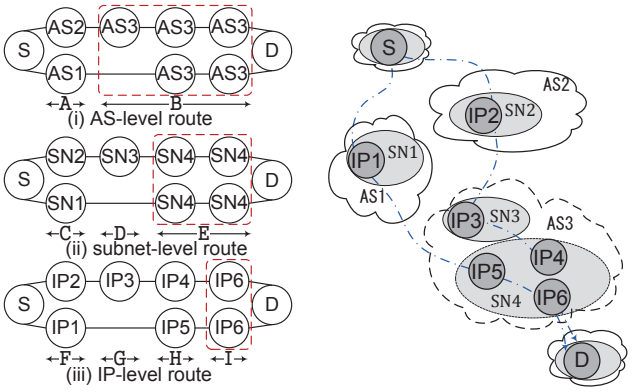
We define the same notations for route $y = Y_1Y_2\cdots Y_{|y|}$, and for any X_i in x and any Y_j in y we can compare them on any level L_t to determine if $L_t(X_i) = L_t(Y_j)$ holds. Given x, y, and $\mathcal{L}$, our problem is to find the differences Δ between x and y via a sequence match, where Δ is the union of n disjoint sets: $\Delta_1 \cup \cdots \cup \Delta_n$. Moreover, the set $\Delta_t, t > 1$, contains the nodes from x and y that are different on L_t (therefore different on any L_s where $s > t$) but identical on L_{t-1} (therefore identical on any L_s where $s < t$); the set Δ_1 contains the nodes that are different on L_1. For the AS-/subnet-/IP-level setting, we have $\Delta = \Delta_{AS} \cup \Delta_{SN} \cup \Delta_{IP}$.

A. Current Approaches

Current approaches all contrast x and y on each level independently. To obtain Δ_{AS}, some previous works (e.g., [39], [37], [10]) employ Jaccard Distance (JD) to count the number of discrepant ASes between two AS-level sequences $AS(x) = AS(X_1)\cdots AS(X_{|x|})$ and $AS(y) = AS(Y_1)\cdots AS(Y_{|y|})$; others (e.g., [43], [42], [8]) employ Edit Distance (ED) to compute the minimum number of deletion, insertion, and replacement operations to equalize $AS(x)$ and $AS(y)$. To obtain Δ_{SN} (Δ_{IP}), JD or ED is similarly performed on $SN(x)$ and $SN(y)$ ($IP(x)$ and $IP(y)$). We agree with the argument by Schwartz et al. [42] that ED is better than JD because it captures the nodes' order. Therefore, we will only focus on ED in our discussions, although the redundancy shown later also exists in JD for a similar reason. Moreover, contrasting two routes with ED is equivalent to computing their longest common subsequence (LCS) with a simple tweak, because $ED(x, y) = |x| + |y| - 2|LCS(x, y)|$ holds by doubling the weight of replacement operations [7]. The only difference is that LCS not only quantifies (i.e., how many nodes are different) but also identifies (i.e., which are the different nodes) the differences. Therefore, our ensuing discussions encompass both the approaches of ED and LCS.

Fig. 1 depicts two routes from S to D (Fig. 1(b)) and their three-level labels (Fig. 1(a)). Current approaches first compute the LCS (equivalently the ED) between $AS(x)$ and $AS(y)$, and identify nodes in segment `A` (segment `B`) as the AS-level differences (similarities): $\Delta_{AS} = \{$AS1-2$\}$. Note that, following existing work [23], nodes in the same AS (e.g., the three/two AS3 nodes for the upper/lower route) are collapsed into the same AS-level entity before the LCS computation. That is, the LCS contrasts $AS(x) = AS2 - AS3$ and $AS(y) = AS1 - AS3$.

They then contrast $SN(x)$ and $SN(y)$, and identify nodes in segments `C` and `D` (segments `E`) as the SN-level differences (similarities): $\Delta_{SN} = \{$SN1-3$\}$. Note also that similar collapsing is applied to SN-level nodes as well (e.g., the two SN4 nodes for both routes) before the LCS computation. That is,



(a) Independent levels. (b) Integrative levels.

Fig. 1. Contrasting two multi-level routes on independent and integrative levels.

$SN(x) = SN2 - SN3 - SN4$ and $SN(y) = SN1 - SN4$. However, segment C is just a duplication of A which has already been accounted for on AS-level comparisons. Finally, they contrast $IP(x)$ and $IP(y)$, and identify nodes in segments F, G, and H (segment I) as the IP-level differences (similarities): $\Delta_{IP} = \{\text{IP1-5}\}$. However, segment F duplicates C, A, and segment G duplicates D.

Therefore, contrasting segment A on SN-/IP-level and segment D on IP-level is redundant and prolongs the analysis. Moreover, the resultant Δ_{AS}, Δ_{SN}, and Δ_{IP} are not disjoint, as some node discrepancy has been duplicated into multiple levels. As a result, the levels entangle to make the intended "three-level" contrasting ambiguous. Algorithm 1, named cmp, presents the general algorithm for contrasting two n-level routes by current approaches.

Algorithm 1 cmp(rt_pair,$\mathcal{L}$); rt_pair=$\{x,y\}$ and $\ell \in \{1, \cdots, n\}$; output: $\Delta = \Delta_1 \cup \cdots \cup \Delta_n$

```
1: for all L_ℓ ∈ ℒ do
2:     (lcs_segs,non_lcs_segs) ← LCS(rt_pair,L_ℓ);
3:     Δ_ℓ ← Δ_ℓ ∪ non_lcs_segs;
4: end for
```

We observe that the redundancy comes from the *fixed resolutions* used for contrasting the entire routes, while different resolutions should have been applied for different route segments. For example, none of the three fixed resolutions is the best for contrasting the entire routes depicted in Fig. 1. More specifically, the AS-level resolution is the best for identifying the degree of difference in segment A, because lower levels are too granular to see that they actually go through distinct ASes. But segment D is accurately characterized only by the SN-level resolution, because the AS-level resolution is too coarse to capture any difference, and the IP-level too fine to see the traversals through different SNs. Segments H and I, on the other hand, require the IP-level resolution, because no other resolution can expose such low-level node traversal. This insight reveals that the redundancy can be eliminated by choosing the fittest resolution for each segment.

B. An Efficient Approach

We eliminate the redundancy by a *just-enough-resolution* approach called rtd (i.e., route-diff). rtd always starts from the AS-level resolution. If the AS-level resolution already identifies the differences in a segment (e.g., for non-LCS portion such as segment A), rtd will not increase its resolution to contrast this segment further. Therefore, the redundant comparisons are avoided. But if the AS-level resolution fails to identify any differences in a segment (e.g., segment B), rtd will switch to the SN-level resolution for that segment to capture possible differences there. Contrasting with the SN-level resolution is also based on LCS, so the resolution switch is implemented by recursively contrasting current-level LCS segments on a finer level. Note that if any AS-level LCS node went through the aforementioned "node collapsing", it will be restored into an SN-level sequence before the SN-level LCS computation. Finally, the recursion returns when it finishes contrasting with the IP-level resolution. In this way, rtd gradually finds the best resolution for each segment and avoids redundant comparisons. Algorithm 2 presents the general algorithm for contrasting two n-level routes with rtd.

Fig. 2 compares the efficiency of cmp and rtd by using the example routes in Fig. 1 as input, and then drawing their "comparison density graphs". For the sake of clarity, we use nodes 1-6 to denote the nodes with labels IP1-6 in Fig. 1, respectively; two nodes are connected by a w-weighted link if they are compared w times by cmp or rtd. Note that when multiple nodes are collapsed into a single entity and compared as a whole, we assign the link(s) to the first node. For example, nodes 3, 4, 6 are collapsed into AS3 on AS-level and we assign the link(s) to node 3; nodes 4, 6 are collapsed into SN4 on SN-level and we assign the link(s) to node 4. We can see that 12 out of 22 comparisons made by cmp are redundant, therefore eliminated by rtd. Moreover, the Δ sets computed by rtd, unlike by cmp, are disjoint, achieving an adequate three-level contrasting. For the example paths, we have $\Delta_{AS} = \{\text{AS1-2}\}$, $\Delta_{SN} = \{\text{SN3}\}$, and $\Delta_{IP} = \{\text{IP4-5}\}$.

Algorithm 2 rtd(rt_pair,ℓ); $\ell \in \{1, \cdots, n\}$ and $L_\ell \in \mathcal{L}$; initially ℓ=1 and rt_pair=$\{x,y\}$; output: $\Delta = \Delta_1 \cup \cdots \cup \Delta_n$

```
1: (lcs_segs,non_lcs_segs) ← LCS(rt_pair,L_ℓ);
2: Δ_ℓ ← Δ_ℓ ∪ non_lcs_segs;
3: if ℓ == n then
4:     return
5: end if
6: for all lcs_seg ∈ lcs_segs do
7:     rtd(lcs_seg,ℓ+1);
8: end for
```

rtd is more efficient than cmp because it eliminates some nodes on each level, and decreases the number of comparisons as the recursion goes deeper. cmp, on the other hand, compares the same set of nodes repetitively on all levels. Let the time complexity of cmp be $T(n,r) = \sum_{i=1}^{n} F(i,r)$, where r

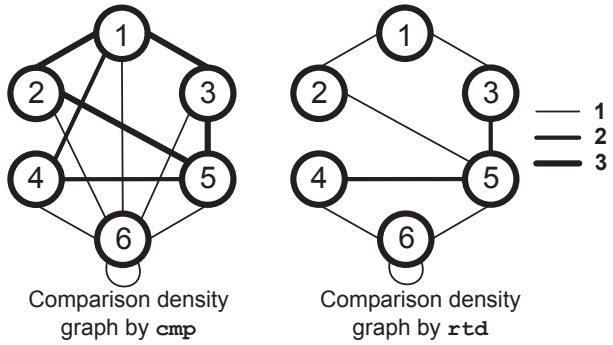


Fig. 2. Comparison density graphs for the example paths: a more densely connected graph results from a more inefficient algorithm.

denotes the average route length and $F(i,r)$ the number of comparisons made by LCS computation on the i-th level. If we solve LCS with the dynamic programming approach[1], we have $F(i,r) = O(r^2)$ regardless of i. Therefore, we have $T(n,r) = O(nr^2)$ for an n-level analysis.

Using a similar set of notations, the time complexity of `rtd` can be written as $T'(n,r) = \sum_{i=1}^{n} F'(i,r)$. When $i = 1$, we have $F'(1,r) = O(r^2)$ because LCS needs to compare the entire r-lengthed routes. However, the first level comparison eliminates all non-LCS nodes so the second level only compares a subset of nodes. We assume a reduction ratio of α_1, $\alpha_1 \in [0..1]$, meaning that the first level LCS has the length of $\alpha_1 r$. Then we have

$$F'(2,r) = O((\alpha_1 r)^2) = \alpha_1^2 O(r^2).$$

More generally, assuming a reduction ratio of α_t from the t-th level to the $(t+1)$-th level, where $t \in [1..n-1]$, we have for the i-th level:

$$F'(i,r) = O((\alpha_1 \cdots \alpha_{i-1} r)^2) = (\alpha_1 \cdots \alpha_{i-1})^2 O(r^2).$$

Therefore, we sum up all the levels to get

$$\begin{aligned} T'(n,r) &= \sum_{i=1}^{n} F'(i,r), \\ &= (1 + \alpha_1^2 + (\alpha_1\alpha_2)^2 + \cdots (\alpha_1 \cdots \alpha_{n-1})^2) O(r^2), \\ &\leq \underbrace{(1 + \ldots + 1)}_{\text{n } \times \text{1's}} O(r^2), \\ &= O(nr^2), \\ &= T(n,r). \end{aligned} \tag{1}$$

Moreover, the equality holds only when $\alpha_1 = \cdots = \alpha_{n-1} = 1$, meaning that x and y are identical on the lowest level L_n (therefore on all higher levels) so that no node can be eliminated on any level. But we can pre-verify this case by performing a node-wise comparison between x and y on the L_n level, which takes $O(r)$ time, and only proceed to call `rtd` if $L_n(x) \neq L_n(y)$. Therefore, we have strictly $T'(n,r) < T(n,r)$, implying that `rtd` always outperforms `cmp` for multi-level route analytics. Furthermore, under an ideal case where $\alpha_1 = \cdots = \alpha_{n-1} = \alpha$, with α being a constant in $[0..1)$, we have

$$\begin{aligned} T'(n,r) &= (1 + \alpha^2 + \alpha^4 + \cdots \alpha^{2n-2}) O(r^2), \\ &= \frac{1-\alpha^{2n}}{1-\alpha^2} O(r^2), \\ &= O(r^2), \end{aligned} \tag{2}$$

because a complexity analysis sets both n and r to infinity. Therefore, `rtd` maintains its efficiency in an n-level analysis by performing $O(r^2)$ comparisons regardless of n.

C. Discussion

`rtd`'s lower time complexity has two implications. For time-critical tasks, `rtd` decreases the computational load so that route analyses finish sooner. For deadline-based tasks, on the other hand, `rtd` downsizes the budget of computing resources (e.g., rented VMs in the cloud) while respecting the same deadline. Consider an n-level route analysis task, where each level of analysis requires M rented VMs to finish before the deadline D. `cmp` has to rent $n \times M$ VMs, but `rtd` meets the deadline D using only M VMs (ideally). Due to space limitation, we focus on evaluating the computational reduction in §III, but a budget reduction is viable for the same reason.

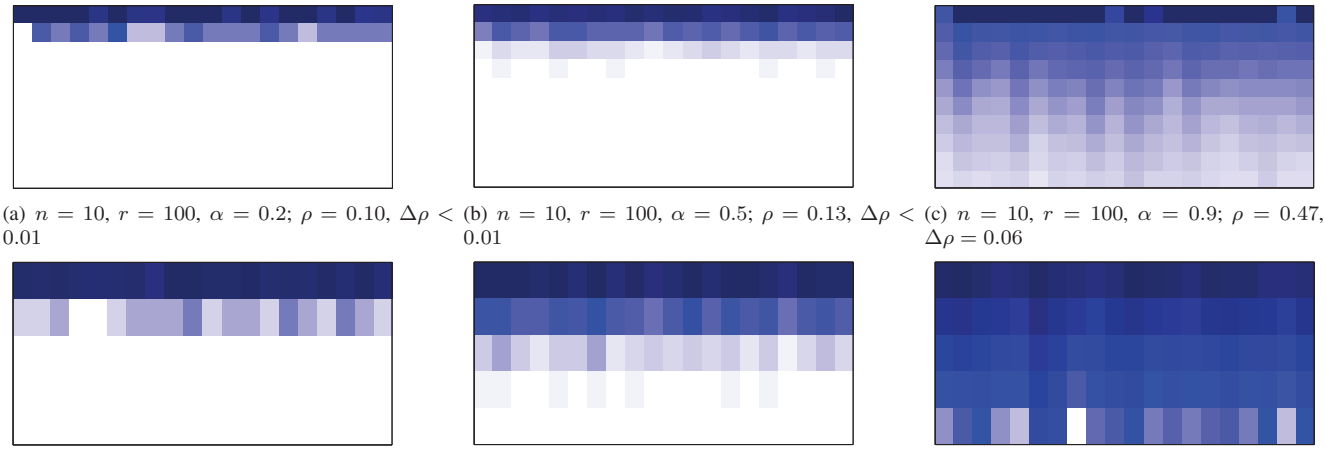
`rtd` and `cmp`, like many other sequence aligning algorithms, can be tuned to output multiple alignments should they exist [40]. For example, routes $A-B-C$ and $A-C-B$ have two LCSes: $A-B$ and $A-C$. However, their application in route contrasting does not require them to do so, because the *quantification* of route differences, either by ED or LCS, is consistent for all alignments. This is because both the ED and the LCS approaches expose the maximum degree of similarity between two routes (e.g., for routes $A-B-C$ and $A-C-B$, both $A-B$ and $A-C$ expose two node matches). Moreover, different implementations of the LCS algorithm will find the same longest common subsequence, if only one were to be used, as long as they scan the routes with the same order.

III. Evaluation

We first evaluate `rtd` on simulated traces with controlled parameters r, n, and α. To this end, we define a "comparison density matrix" $\mathcal{C}_{n \times R}$ when contrasting two n-level routes with length r, where $R = r^2 + r$. We map the t-th level to the t-th row, and each node pair (X_i, Y_j) to a unique column s, where $s = i \cdot r + j$. If X_i and Y_j are compared on the t-th level, we set the entry $c_{t,s}$ to one; otherwise we set it to zero. Therefore, the sum of all entries $\sum_{i=1}^{n}\sum_{j=1}^{R} c_{i,j}$ equals to the number of comparisons in total. Following the notations in §II, we write `rtd`'s matrix as $\mathcal{C}'$ and `cmp`'s $\mathcal{C}$. We visualize the "heatmap" of $\mathcal{C}'$ by aggregating its columns by a bin width of 500; a darker-colored bin's node pairs undergo more comparisons. We do not visualize that of $\mathcal{C}$, because all its rows simply repeat $\mathcal{C}$'s first row.

We vary n from 2 to 100 with a step of 1, r from 10 to 1000 with a step of 10, α from 0.01 to 0.99 with a step of 0.01,

[1] In certain scenarios LCS can be computed with lower complexity [7]. But `rtd` improves the complexity in a similar way.



(a) $n = 10$, $r = 100$, $\alpha = 0.2$; $\rho = 0.10$, $\Delta\rho < 0.01$ (b) $n = 10$, $r = 100$, $\alpha = 0.5$; $\rho = 0.13$, $\Delta\rho < 0.01$ (c) $n = 10$, $r = 100$, $\alpha = 0.9$; $\rho = 0.47$, $\Delta\rho = 0.06$

(d) $n = 5$, $r = 100$, $\alpha = 0.2$; $\rho = 0.21$, $\Delta\rho < 0.01$ (e) $n = 5$, $r = 100$, $\alpha = 0.5$; $\rho = 0.26$, $\Delta\rho < 0.01$ (f) $n = 5$, $r = 100$, $\alpha = 0.9$; $\rho = 0.68$, $\Delta\rho < 0.01$

Fig. 3. Heatmaps of comparison density matrices by `rtd`; darkest (lightest) bins contain 200 (0) comparisons.

and run simulations for each setting. Due to space limitation, Fig. 3 only presents six settings' heatmaps to show the level-by-level reduction. We can see that the uppermost regions are the darkest (i.e., the highest levels processed almost the entire routes), and the colors grow lighter for lower regions (i.e., less nodes were left for lower levels). We also compute the comparison reduction ratio achieved by `rtd` over `cmp` for each setting:

$$\rho = (\sum_{i=1}^{n}\sum_{j=1}^{R} c'_{i,j})/(\sum_{i=1}^{n}\sum_{j=1}^{R} c_{i,j}), \tag{3}$$

where $c'_{i,j}$ ($c_{i,j}$) is an entry from $\mathcal{C}'$ ($\mathcal{C}$). We can see that even with $\alpha = 0.9, n = 10$ (i.e., each level only reduces 10% nodes), `rtd` still eliminates 53% comparisons in total. When $\alpha = 0.2/0.5$, `rtd` manages to eliminate all nodes even before the lowest level is reached. We further compare the achieved ρ with its theoretical value ρ^* by computing $\Delta\rho = |\rho - \rho^*|$, where

$$\rho^* = \frac{T'(n,r)}{T(n,r)} = \frac{1-\alpha^{2n}}{(1-\alpha^2)n}. \tag{4}$$

We can see that all six ρ's fit their expected values quite well. For a more comprehensive view, Fig. 4 plots the relation between ρ and all simulated α and n (we set r=100 for consistency with Fig. 3): `rtd` eliminates 91% comparisons on average. Morever, Eqn. (4) also implies that the reduction is more significant with a smaller α or a bigger n, which can be validated by Figs. 3 and 4.

We then evaluate `rtd` on Ark, iPlane, and FastMapping datasets by contrasting random routes on the same prober, which was employed by Beverly et al. [8] to quantify unnecessary probes. We use $\frac{2}{3}$ of Ark data collected in 2011, $\frac{1}{7}$ of FastMapping data, and 4 rounds of iPlane data collected in April 2012. For each Ark or iPlane monitor, we select random route pairs to compare until a certain pair has been selected twice, and repeat the process for each probing round; for each

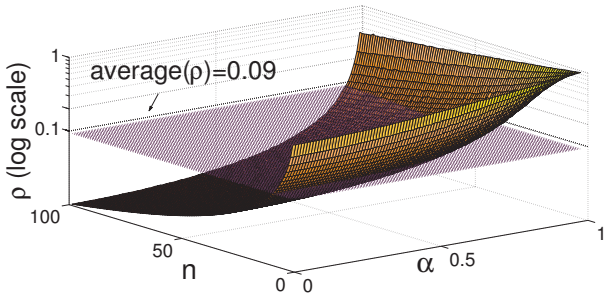
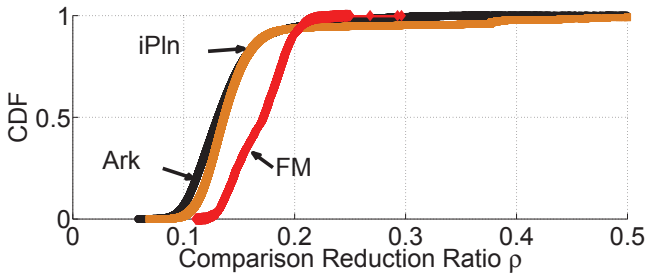


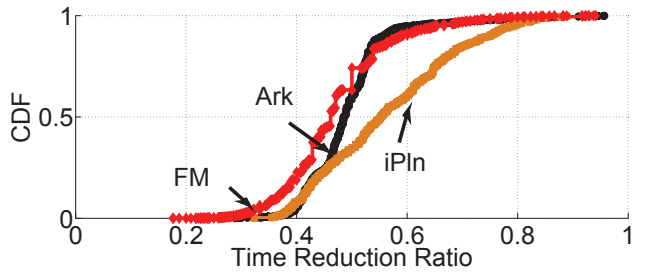
Fig. 4. `rtd` eliminates more comparisons as α grows smaller or n grows larger; $\alpha \in [0.01, 0.99]$, $n \in [2, 100]$, r=100.

FastMapping monitor, we compare all its route pairs in each probing round. We have evaluated 87 million routes and made 52 billion comparisons in total, which are much larger-scale than the previous study [8] that collected one month's data from one Ark monitor and one iPlane monitor, comprising 0.4 million routes. We apply the setting of IP-, /24-, AS-levels [31], where the AS routes are converted from IP routes by analyzing BGP tables. Although such mappings may have their weaknesses [35], we use the same mappings for `rtd` and `cmp` to ensure a fair comparison.

Fig. 5(a) plots the CDF of comparison reduction ratios for each dataset, where each data point aggregates the results for contrasting 5K routes. For Ark, iPlane, and FastMapping datasets respectively, 86%, 85%, and 83% comparisons made by `cmp` are eliminated by `rtd` on average. Moreover, the resemblance of Ark and iPlane CDFs is due to their similar measurement settings: Ark probes all /24 prefixes and iPlane all BGP prefixes. Such settings resulted in similar topological properties, such as Internet coverage and sizes of traversed ASes. FastMapping, however, only probes 1K destinations which resulted in a notably discrepant CDF.

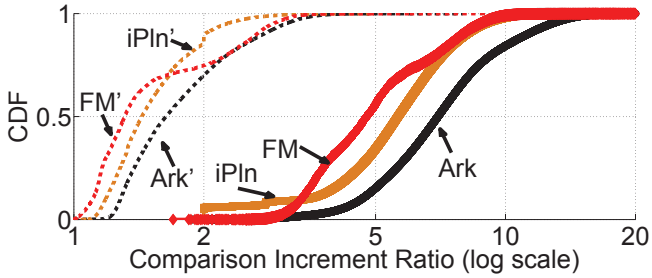


(a) Reduction ratio of node comparisons by `rtd`.

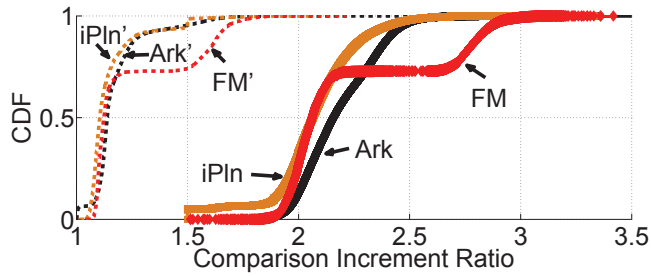


(b) Reduction ratio of running time by `rtd`.

Fig. 5. `rtd` on average eliminate 85% comparisons and halves the running time in the IP/24-prefix/AS setting.



(a) The increment of comparisons from one level to two levels.



(b) The increment of comparisons from two levels to three levels.

Fig. 6. The number of comparisons increases more slowly with the number of levels in `rtd` (labeled as Ark', iPln', FM') than in `cmp` (labeled as Ark, iPln, FM).

We also log the running time of `rtd` and `cmp` on a Linux machine with a 2.13GHz CPU and a 8GB memory. Fig. 5(b) plots the CDFs of `rtd`'s acceleration ratios, where each data point aggregates the results for contrasting 6 million routes. `rtd` reduces the running time to 49%, 56%, and 47% on average for Ark, iPlane, and FastMapping datasets, respectively, and achieves a speed of 63K route contrasting operations per second. The average time reduction ratio over three datasets (51%) is less pronounced than the average comparison reduction ratio (85%), which is expected because node comparisons are not the only source of computation. Other sources include LCS construction, state memoization, and recursions, and the cost of the first two factors generally grows with route lengths. Therefore, the time reduction is the most (least) significant for FastMapping (iPlane) routes that have the shortest (longest) average length of 11 (15) hops. Another reason for the higher efficiency achieved on FastMapping is that its sparser coverage results in more diverse AS routes than in other datasets, which further results in a higher AS-level elimination ratio. `rtd` therefore spends less time performing recursions into deeper levels.

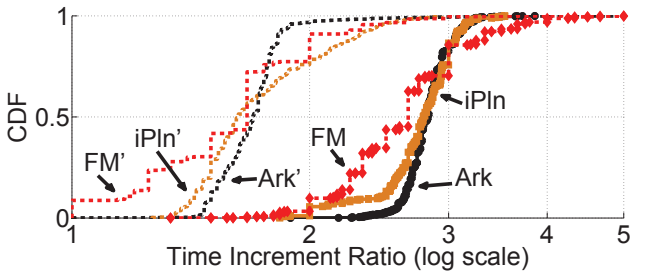
`rtd`'s efficiency over `cmp` (i.e., Eqn. (1)) also implies that the number of comparisons increases more slowly with the number of levels in use. To evaluate this property, we compute $\sum_{i=1}^{n}\sum_{j=1}^{R} c'_{i,j}$ and $\sum_{i=1}^{n}\sum_{j=1}^{R} c_{i,j}$ for one-level (i.e., AS), two-level (i.e., AS/SN), and three-level (i.e., AS/SN/IP) settings, and count how many more comparisons are incurred by each addition level (Fig. 6). For Ark, iPlane, and FastMapping datasets, adding the SN-level to the AS-level incurs 6.9, 5.6, 4.8 times (medians) comparisons by `cmp`, but only 1.6, 1.4, 1.2 times by `rtd`, respectively. Adding the IP-level to the AS-/SN-levels further incurs 2.2, 2.1, 2.1 times comparisons by `cmp`, but only 1.1, 1.1, 1.1 times by `rtd`. The increment from the AS-level to the AS-/SN-levels is larger than that from the AS-/SN-levels to the AS-/SN-/IP-levels, because the SN-level routes are much longer than the AS-level routes and therefore add many more nodes, but the IP-level routes add relatively less nodes to the existing ones on the AS-/SN-levels.

Moreover, Fig. 7 plots the CDFs of running time increment ratios. The running time from one level to two levels, averaged over three datasets, is increased to 2.8 times by `cmp` but only 1.7 times by `rtd`, and that from two levels to three levels is increased to 1.7 times by `cmp` but only 1.5 times by `rtd`. The efficiency achieved on FastMapping is still higher due to the reasons said before.

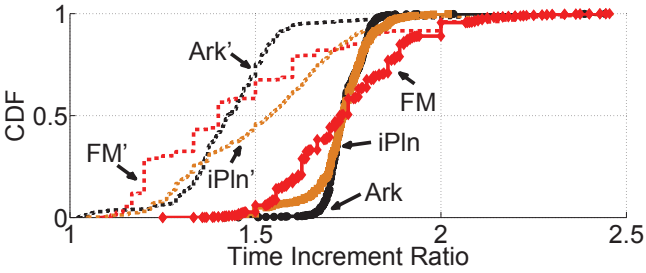
IV. A Route Clustering Application

In this section, we design a practical route clustering application using `rtd`, and apply it to monitor an ISP transition. Clustering is a popular technique in exploratory data analysis [28], but its computational efficiency over big datasets poses a key challenge [48]. A hierarchical clustering of m routes, for example, requires $O(m^2)$ contrasting operations between all route pairs to construct a proximity matrix. Route contrasting operations are therefore expected to take no more time than necessary. As the first route clustering study that we know of, we demonstrate that clustering offers insightful results, and that it is computationally feasible for route dynamics studies whose m's are moderate. Our results also encourage more research on high-performance analytics that may eventually enable route clustering with Internet-scaled m's.

We cluster the `tcptraceroute` [46] data on a path from Hong Kong to Israel that was sampled by our monitor [11]



(a) The increment of running time from one level to two levels.



(b) The increment of running time from two levels to three levels.

Fig. 7. The running time increases more slowly with the number of levels in `rtd` than in `cmp`.

once in two minutes from 26 Feb. to 18 Mar. 2010, and study the route evolution during our ISP transition with two network switches. We employ an agglomerative, complete-linkage clustering and a weighted Minkowski distance measure $d(x,y) = w_{AS} \cdot |\Delta_{AS}| + w_{SN} \cdot |\Delta_{SN}| + w_{IP} \cdot |\Delta_{IP}|$ [48], with $w_{AS}/w_{SN}/w_{IP} = 9/7/2$. This weighting emphasizes higher-level route changes that potentially have more impact, although determining the "best" weighting, if any, is out of our scope. We normalize it to $[0..1]$ by computing

$$d'(x,y) = \frac{d(x,y)}{w_{AS} \cdot (|x| + |y|)}.$$

Finally, we apply Davies-Bouldin Index [14], which balances intra-cluster closeness and inter-cluster separation [36], to select the optimal clustering hierarchy with three clusters.

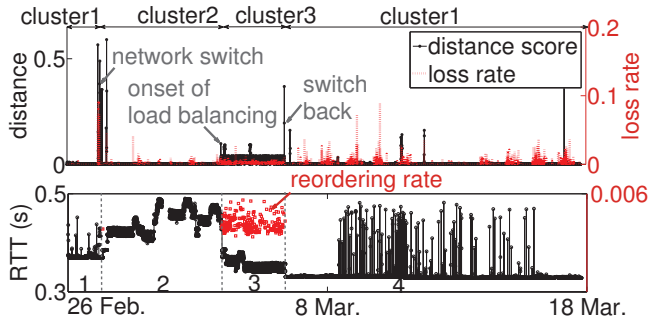


Fig. 8. Route clusters and their performance features.

Fig. 8 annotates the route evolution with the performance metrics measured by OneProbe [32]; the upper figure also plots the time series of $d'(x,y)$ between all consecutive route snapshots. The time series detects that the first network switch was one hour behind schedule, which has been confirmed by the operators. Moreover, the original route (i.e., cluster 1) which had a stable RTT baseline with intermittent surges was transitioned to an intermediate route (i.e., cluster 2) with inflated RTTs; the higher delays were introduced by eight extra hops in the new route that detoured from Taiwan to Europe. The load-balancer that transitioned cluster 2 to 3 was set up by an European ISP, and it decongested the route and reduced its RTTs. However, it employed a per-packet splitting and thus induced packet reordering. The second network switch restored both the route and its RTTs to their original patterns (i.e., the second appearance of cluster 1), but the new ISP resulted in a decreased RTT baseline and a clearer diurnal loss pattern. We can see that the four-phased transition annotated by route clustering provides deeper insights into the transitional events. Moreover, clustering the 14K routes collected in 20 days only took one minute on the aforementioned machine, with `rtd`'s pre-verification turned on. Such efficiency makes clustering analysis feasible for route dynamics studies (e.g., [38], [13], [12]), which typically maintain a moderate range of destinations (e.g., 1K for FastMapping) and a high traceroute frequency. Therefore, we will apply similar analysis to those routes as future work, aiming to expose unlearnt route properties and performance correlations.

V. Related Work

Contrasting multi-level routes underpins many existing network measurement research (e.g., [39], [37], [22], [43], [31], [10], [21]). However, current approaches contrast them on each level independently with Jaccard Distance (e.g., [39], [37], [10]), Edit Distance (e.g., [43], [42]) or their variants (e.g., [22], [21]), therefore incur computational redundancy. Our approach, on the other hand, integrates all levels for redundancy elimination, and improves the analysis efficiency of Internet-sized route data. Studies of route sharing, on multiple levels or not, can also facilitate route stability, prevalence, (e.g., [38], [12]), predictability [13], and similarity [25] measurement, or available bandwidth estimation [24]. Our approach can compensate their analysis by evaluating the information gain provided by multi-level routes over single-level ones, or by adding extra levels (e.g., organizational [9] or geographical [29], [20] levels) to the current multi-level analysis.

Existing research enhances the route-tracing technique in many dimensions. The accuracy of route tracing is improved by Paris-traceroute [4] that handles load-balancers and AS-level traceroute [35], [34] that resolves AS-level sequences. The thoroughness of route tracing is enhanced by MDA traceroute [5] that enumerates parallel routes, TraceNET [47] that explores routes' subnetting structure, reverse traceroute [30] that estimates the backward route, NANOG-traceroute [2] that outputs extra metrics such as the path MTU, IPv6 traceroute [1] that traces the IPv6 route instance, and others. Moreover, route tracing is made more efficient by FastMapping [12] that measures load-balanced paths, DTRACK [13] that reduces probes on stable paths, DoubleTree [18] and its

variants [17], [16] that coordinate distributed monitors, ISC strategy [8] that minimizes redundant probes, just to name a few. Our paper is orthogonal to the aforementioned research, because we enhance route analysis instead of route tracing.

VI. Conclusions and Future Work

We identified that contrasting multi-level routes by current approaches incurs computational redundancy, exposed that the root cause is the independent use of levels, and proposed `rtd` that integrates all levels recursively to eliminate the redundancy. Our extensive evaluations on simulated traces and Ark, FastMapping, iPlane datasets showed that `rtd` eliminates 85% comparisons on average, and doubles the analysis speed. We also demonstrated that route clustering is feasible and insightful for route dynamics studies. In the future, we will evaluate the full clustering capacity of commodity PCs, and explore more efficient contrasting approaches that allow route clustering in larger scales (e.g., Ark datasets), including hardware implementations and distributed computations.

Acknowledgement

This work is partially supported by a grant (ref. no. GHP/027/11) from the Innovation Technology Fund in Hong Kong, a grant (ref. no. H-ZL17) from the Joint Universities Computer Centre of Hong Kong, and a grant (ref. no. G-YK26) from The Hong Kong Polytechnic University.

References

[1] IPv6 Traceroute. http://www.ipv6tools.org.
[2] NANOG Traceroute. http://man.he.net/man8/traceroute-nanog.
[3] D. Agrawal, S. Das, and A. Abbadi. Big data and cloud computing: New wine or just new bottles? *PVLDB Endowment*, 3(2), 2010.
[4] B. Augustin, X. Cuvellier, B. Orgogozo, F. Viger, T. Friedman, M. Latapy, C. Magnien, and R. Teixeira. Avoiding traceroute anomalies with Paris traceroute. In *Proc. ACM/USENIX IMC*, 2006.
[5] B. Augustin, R. Teixeira, and T. Friedman. Measuring load-balanced paths in the Internet. In *Proc. ACM/USENIX IMC*, 2007.
[6] R. Barnes. Mapping the great void: Smarter scanning for IPv6. In *Proc. CAIDA AIMS-4*, 2012.
[7] L. Bergroth, H. Hakonen, and T. Raita. A survey of longest common subsequence algorithms. In *Proc. IEEE SPIRE*, 2000.
[8] R. Beverly, A. Berger, and G. Xie. Primitives for active Internet topology mapping: Toward high-frequency characterization. In *Proc. ACM/USENIX IMC*, 2010.
[9] X. Cai, J. Heidemann, B. Krishnamurthy, and W. Willinger. Towards an AS-to-Organization map. In *Proc. ACM/USENIX IMC*, 2010.
[10] E. Chan, X. Luo, W. Fok, W. Li, and R. Chang. Non-cooperative diagnosis of submarine cable faults. In *Proc. PAM*, 2011.
[11] R. Chang, W. Fok, W. Li, E. Chan, and X. Luo. Neighbor-Cooperative measurement of network path quality. In *Proc. IEEE Globecom*, 2010.
[12] I. Cunha, R. Teixeira, and C. Diot. Measuring and characterizing end-to-end route dynamics in the presence of load balancing. In *Proc. PAM*, 2011.
[13] I. Cunha, R. Teixeira, D. Veitch, and C. Diot. Predicting and tracking Internet path changes. In *Proc. ACM SIGCOMM*, 2011.
[14] D. Davies and D. Bouldin. A cluster separation measure. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 1(2), 1979.
[15] J. Dean and S. Ghemawat. MapReduce: Simplified data processing on large clusters. In *Proc. USENIX OSDI*, 2004.
[16] B. Donnet and T. Friedman. Topology discovery using an address prefix based stopping rule. In *Proc. EUNICE Worshop*, 2005.
[17] B. Donnet, T. Friedman, and M. Crovella. Improved algorithms for network topology discovery. In *Proc. PAM*, 2005.
[18] B. Donnet, P. Raoult, T. Friedman, and M. Crovella. Effcient algorithms for large-scale topology discovery. In *Proc. ACM SIGMETRICS*, 2005.
[19] EMC Corporation. MPP database. http://www.greenplum.com/products/greenplum-database.
[20] B. Eriksson, P. Barford, J. Sommers, and R. Nowak. A learning-based approach for IP geolocation. In *Proc. PAM*, 2010.
[21] J. Han, D. Watson, and F. Jahanian. An experimental study of Internet path diversity. *IEEE Trans. Dependable and Secure Computing*, 3(4), 2006.
[22] Y. He, M. Faloutsos, and S. Krishnamurthy. Quantifying routing asymmetry in the Internet at the AS level. In *Proc. IEEE GLOBECOM*, 2004.
[23] Y. He, M. Faloutsos, S. Krishnamurthy, and B. Huffaker. On routing asymmetry in the Internet. In *Proc. IEEE GLOBECOM*, 2005.
[24] N. Hu and P. Steenkiste. Exploiting Internet route sharing for large scale available bandwidth estimation. In *Proc. ACM/USENIX IMC*, 2005.
[25] N. Hu and P. Steenkiste. Quantifying Internet end-to-end route similarity. In *Proc. PAM*, 2006.
[26] Y. Hyun. Archipelago measurement infrastructure. http://www.caida.org/projects/ark/.
[27] V. Jacobson. Traceroute. ftp://ftp.ee.lbl.gov/traceroute.tar.gz.
[28] A. Jain, M. Murty, and P. Flynn. Data clustering: a review. *ACM Computing Surveys*, 31(3), 1999.
[29] E. Katz-Bassett, J. John, A. Krishnamurthy, D. Wetherall, T. Anderson, and Y. Chawathe. Towards IP geolocation using delay and topology measurements. In *Proc. ACM/USENIX IMC*, 2006.
[30] E. Katz-Bassett, H. Madhyastha, V. Adhikari, C. Scott, J. Sherry, P. van Wesep, T. Anderson, and A. Krishnamurthy. Reverse traceroute. In *Proc. USENIX NSDI*, 2010.
[31] C. Logg, L. Cottrell, and J. Navratil. Experiences in traceroute and available bandwidth change analysis. In *Proc. ACM SIGCOMM Workshop on Network Troubleshooting*, 2004.
[32] X. Luo, E. Chan, and R. Chang. Design and implementation of TCP data probes for reliable and metric-rich network path monitoring. In *Proc. USENIX Annual Tech. Conf.*, 2009.
[33] H. Madhyastha, T. Isdal, M. Piatek, C. Dixon, and T. Anderson. iPlane: An information plane for distributed services. In *Proc. USENIX OSDI*, 2006.
[34] Z. Mao, D. Johnson, J. Rexford, J. Wang, and R. Katz. Scalable and accurate identification of AS-level forwarding paths. In *Proc. IEEE INFOCOM*, 2004.
[35] Z. Mao, J. Rexford, J. Wang, and R. Katz. Towards an accurate AS-level traceroute tool. In *Proc. ACM SIGCOMM*, 2003.
[36] U. Maulik and S. Bandyopadhyay. Performance evaluation of some clustering algorithms and validity indices. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 24(12), 2002.
[37] A. Pathak, H. Pucha, Y. Zhang, Y. Hu, and Z. Mao. A measurement study of Internet delay asymmetry. In *Proc. PAM*, 2008.
[38] V. Paxson. End-to-end routing behavior in the Internet. In *Proc. ACM SIGCOMM*, 1996.
[39] H. Pucha, Y. Zhang, Z. Mao, and Y. Hu. Understanding network delay changes caused by routing events. In *Proc. ACM SIGMETRICS*, 2007.
[40] C. Rick. Efficient computation of all longest common subsequences. In *Proc. SWAT*, 2000.
[41] SAS Institute, Inc. Solutions for high-performance analytics. http://www.sas.com/software/high-performance-analytics/.
[42] Y. Schwartz, Y. Shavitt, and U. Weinsberg. A measurement study of the origins of end-to-end delay variations. In *Proc. PAM*, 2010.
[43] Y. Schwartz, Y. Shavitt, and U. Weinsberg. On the diversity, stability and symmetry of end-to-end Internet routes. In *Proc. IEEE GI Symposium*, 2010.
[44] N. Spring, R. Mahajan, and D. Wetherall. Measuring ISP topologies with Rocketfuel. In *Proc. ACM SIGCOMM*, 2002.
[45] Teradata Inc. Big data analytics database. http://www.teradata.com/m/Solutions/Big-Data-Analytics/Database/.
[46] M. Toren. Tcptraceroute. http://michael.toren.net/code/tcptraceroute/.
[47] M. Tozal and K. Sarac. TraceNET: An Internet topology data collector. In *Proc. ACM/USENIX IMC*, 2010.
[48] R. Xu and D. Wunsch. Survey of clustering algorithms. *IEEE Trans. Neural Netw.*, 16(3), 2005.
[49] W. Zhou, Q. Li, M. Caesar, and P. Godfrey. ASAP: A low-latency transport layer. In *Proc. ACM CoNEXT*, 2011.