Orca-SR: A Real-Time Traffic Engineering Framework leveraging Similarity Joins

Jees Augustine[‡], Suraj Shetiya[‡], Abolfazl Asudeh[†], Saravanan Thirumuruganathan[§], Azade Nazi[¶], Nan Zhang[*], Gautam Das[‡], Divesh Srivastava[‖]

[‡]University of Texas at Arlington; [†]University of Illinois at Chicago; [§]QCRI, HBKU; [¶]Google Brain; [*]School of Business, American University; [‖]AT&T Labs-Research

[‡]{jees.augustine@mavs,suraj.shetiya@mavs,gdas@cse}.uta.edu, [†]asudeh@uic.edu, [§] sthirumuruganathan@hbku.edu.qa, [¶]azade@google.com, [*]nzhang@american.edu, [‖] divesh@research.att.com

ABSTRACT

Reconstructing a high dimensional unknown signal, using lower dimensional observations is a challenging problem, known as *signal reconstruction problem* (SRP), with diverse applications including network traffic engineering, medical image reconstruction, and astronomy. Recently the database community has shown significant advancements in solving the SRP problem efficiently, effectively, and in scale by leveraging database techniques such as similarity joins. In this demo, we demonstrate ORCA-SR that highlights the benefits of signal reconstruction in scale by demonstrating real-time network traffic flow analysis on large networks that were not possible before. ORCA-SR is a web application that enables a user to generate network flow and load the network for interactive analysis of the impact of different traffic patterns on signal reconstruction.

PVLDB Reference Format:
Jees Augustine, Suraj Shetiya, Abolfazl Asudeh, Saravanan Thirumuruganathan, Azade Nazi, Nan Zhang, Gautam Das, Divesh Srivastava. Orca-SR: A Real-Time Traffic Engineering Framework leveraging Similarity Joins. *PVLDB*, 13(12): 2977 - 2980, 2020.
DOI: https://doi.org/10.14778/3415478.3415523

1. INTRODUCTION

Developing scalable algorithms for processing and analyzing massive data systems have been the mainstays of database research for many decades. Database optimization techniques, in particular, have been applied beyond the realm of the database community to yield scalable algorithms for solving problems in the areas of astronomy, computer networks, machine learning, etc.
Signal Reconstruction Problem (SRP) is a challenging problem with diverse applications where the objective is to reconstruct a high dimensional unknown signal using lower dimensional observations. Prominent database techniques such as similarity joins and selectivity estimation for set similarity queries, can yield significant performance improvements on SRP by optimizing for the computational bottlenecks. Several algorithms were proposed in [3] and its extended version [2] for a scalable solution for SRP problem. One popular application of SRP is in Computer Networks, where the observation of a system is limited[1] to the edge level traffic due to infrastructural limitations. We demonstrate ORCA-SR, a system for traffic flow estimation, which can estimate source-destination traffic flows in *real-time*.
Signal Reconstruction Problem (SRP): The aim of SRP is to solve a linear system of the form $AX = b$, where $X \in \mathbb{R}^m$ is a high-dimensional unknown *signal*, $b \in \mathbb{R}^n$ ($n \ll m$) is a low-dimensional projection of X that can be observed in practice and A is a $n \times m$ matrix that captures the linear relationship between X and b.
Traffic Matrix Computation as SRP: Consider an IP network with r ingress/egress points where network traffic may enter or leave with any of the $m = r^2$ source-destination flows (SD flow). Knowing the traffic flow between any source-destination pair is crucial in traffic engineering. Considering the infrastructural limitations, one usually cannot directly measure SD flows. Protocols such as SNMP, collect the link-level flow information, however, falls short in estimating the end-to-end traffic flow. Hence, SRP seeks to estimate the flow between every SD pairs from link-level flow information and routing policy available through routing protocols like BGP. With the growing network size (a few billion SD pairs), prior solutions [8, 9] for SRP simply do not scale. Hence, we proposed methods that support *large scale signal reconstruction* [3, 2]. ORCA-SR leverages these methods and demonstrates a real-time system for interactive network traffic analysis for helping the network analysts to visualize a selected network along with the reconstructed traffic flows.
Problem Formulation: Computer networks are often represented as graph where nodes corresponds to endpoints while the edges correspond to network links. The interconnection between different SD pairs is defined by a routing policy and is commonly represented as a routing matrix, $\mathcal{A}$. It is a binary matrix with edges as rows and SD pairs

[1]SRP has numerous applications in diverse domains that are elaborated in [1], the report for the SIGMOD Research Highlight 2019.

Proceedings of the VLDB Endowment, Vol. 13, No. 12
ISSN 2150-8097.
DOI: https://doi.org/10.14778/3415478.3415523

as columns where for the edges involved in the route between SD pairs is set to 1. The observable properties of the network include the flow at an edge, known as edge traffic vector, b and $\mathcal{A}$. The traffic reconstruction problem is defined as resolving individual SD traffic, $\mathcal{X}$ - represented as a vector, when routing matrix $\mathcal{A}$ and edge traffic vector b are available. This linear system of equations is represented as,

$$\mathcal{A} \cdot \mathcal{X} = b$$

where $\mathcal{A} \in \mathbb{Z}_2^{n \times m}$ is a sparse binary matrix, where n is number of edges in the network and m the number of SD pairs($n \ll m$).

SRP could not be solved using system of linear equation techniques as the number of variables m is larger than the number of equations, n, rendering the system underdetermined with no unique solution. Hence, an additional criterion is needed to choose a distinct $\mathcal{X}$ from the infinitely many possible solutions. A common secondary criterion used in signal reconstruction is the use of a prior point, $\mathcal{X}'$, provided by experts [9]. The final traffic flow values are obtained by taking the point in the solution space that is closest to the given prior. In [3], we provide a closed form formula

$$\mathcal{X} = \mathcal{X}' - \mathcal{A}^T(\mathcal{A}\mathcal{A}^T)^{-1}(\mathcal{A}\mathcal{X}' - b) \quad (1)$$

As evident from Equation 1, $\mathcal{A}\mathcal{A}^T$, is the computational bottleneck for the algorithm which runs in $\mathcal{O}(n^2 m)$. It turns out that each diagonal element $(\mathcal{A}\mathcal{A}^T)[i][i]$ is the upper bound for values in its i^{th} row and i^{th} column of $\mathcal{A}\mathcal{A}^T$. Using techniques from set similarity joins [4] and selectivity estimation for similarity queries [6], we can improve the performance of the algorithm. We now briefly describe the three key ideas. Please refer to [3, 2] for comprehensive details.

- *Thresholding*: We use the diagonal values as the threshold for the values on the same row/column and prune the ones that are below a threshold τ.
- *Conversion to Set Operations*: We transform the problem of computing the inner product between the rows $\mathcal{A}[i]$ and $\mathcal{A}[j]$ (for computing $\mathcal{A}\mathcal{A}^T[i,j]$) into a set similarity instance.
- *Similarity Joins*: We design a hybrid algorithm combining the threshold-based and sketch-based similarity estimation to reduce the complexity of computing a cell in $\mathcal{A}\mathcal{A}^T$ from $O(m)$ to $O(\log m)$.

The exact version of DIRECT algorithm based on Eqn 1 is referred to as DIRECT-E and the approximate version leveraging database optimizations proposed above forms DIRECT-A.

2. ORCA-SR

In this section we describe ORCA-SR, its architecture and implementation details. We describe various technical challenges and user interaction with ORCA-SR.

2.1 Architecture

ORCA-SR is a web application where users submit a query for generating the network, along with the parameters for generating the traffic and analyzing it. Architecture of ORCA-SR is shown in Figure 1. Users submit their requests through a web browser, to ORCA-SR Web Server. These include network type, edge level traffic distribution, a prior probability distribution for traffic and, a threshold. The ORCA-SR web

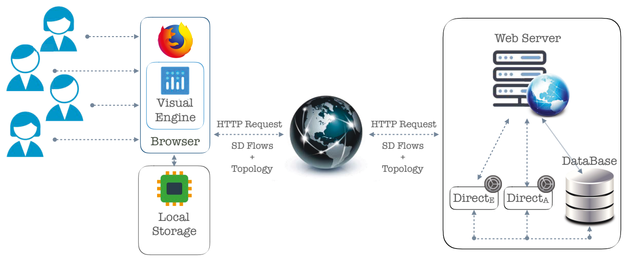


Figure 1: Architecture of Orca-SR

Server generates the routing matrix and traffic vector and subsequently applies the `SRP` algorithm (described in [3]), either DIRECT-E or DIRECT-A based on users requirement to generate exact or approximate answers. Once the algorithm finds SD pair traffic flows, the server sends the results back to the client's visual component which processes and renders the flows. Local storage at client browser caches the SD pairs flow information to answer interactive queries by users. With ORCA-SR, a user can learn more about the applied optimizations by choosing a small subset of the network and subsequently focusing on the sub-network and, changing its input parameters. One can also fine-tune the parameters interactively through our visual interface.

Our application uses the exact algorithm DIRECT-E and the approximated one DIRECT-A for scalability, both implemented in C++ . For orchestrating the Web App, we use Shiny from R[2]. For visual rendering at client browser we use Plotly[3] which internally uses the visualization library D3.js.

2.2 Technical challenges

In this subsection, we discuss major challenges encountered, their resolutions, and the reasoning for their choices. **Scalability through Parallelization**: The first challenge entails from the large scale of the network that we simulate. A general assumption in [3, 2] is the availability of $\mathcal{A}$, a matrix at the scale of up to hundreds of millions SD pairs. Materializing $\mathcal{A}$ at this point is beyond the capabilities of a Web application. Even the sparse representation of $\mathcal{A}$ is in the order of a few 100 GBs. Hence, we designed optimized binary matrix multiplication libraries that operate on the sparse matrix for the implementation of DIRECT-E in C++, which utilizes the underlying processor parallelism. These modules take advantage of the underlying parallelism in processors. And, each of the multiplication will have a depth of the length of path for each source destination pair in $\mathcal{A}\mathcal{A}^T$ multiplication. If the mean length of the path in an un-directed path will be $l = \frac{1}{n \times (n-1)} \sum_{i \neq j} d_{ij}$ where n is the number of distance nodes in the network and this approach provides for a space and performance advantages to algorithms proposed in [3, 2].

During the simulation process, a sequential path generator is the bottleneck for the application. Hence, we designed a parallelized version of the route generator considering the performance issues. Our code can be found at `https://github.com/jeesaugustine/orca_demo`.
Routing: Another major challenge is obtaining the routing matrix $\mathcal{A}$ for large networks. Algorithms proposed in [3, 2]

[2]https://www.rstudio.com/products/shiny/shiny-server/
[3]https://plot.ly/

require a user to furnish $\mathcal{A}$, b and $\mathcal{X}'$ for computing Direct-e and Direct-a. Even with parallelization, assigning the shortest path to billions of SD pairs is computationally expensive. We use a clustering-based approach in addition to the shortest paths to generate and thus $\mathcal{A}$. This is analogous to a hierarchical routing approach [7] practiced in the networking community. Please refer to [3] for further details.

Multi-User Interaction: Algorithms described in [3] were designed for solving a single instance of SRP. To facilitate multiple users interacting with the system seamlessly, we added a string of optimizations such as caching that avoids redundant computation of expensive results and sandboxing that separates the requests of different users.

2.3 User interfaces

Orca-SR has extensive functionalities and has multiple interfaces. Due to the space limitations, we limit ourselves to the key functionalities of Orca-SR in the write up.

Network Graph Simulation - Figure 2 shows the primary input interface. We provide support for various forms of input graphs (i) real networks (from SNAP repository) (ii) a random graph generator by Erdős–Rényi random graph generation[4] (based on user given parameters) and, (iii) a user provided network. Users may choose different approaches for modeling *prior* probability distribution for traffic and various traffic generation methods according to the choice and fitting to the application domain requirements. A user will also get a choice of different thresholds for the Direct-a algorithm as fractions of the number of SD pairs in the network. On selecting a node, Orca-SR will process the request and display the traffic flowing from the selected node to all other nodes in the network. A user may download all the artifacts of the network and flow (as *.csv* files) for further investigation.

SD Flow Analysis - The chosen parameter values are used to calculate the SD traffic for each SD pair. A user can interact with the flow values to selectively view parts of the output, focusing on flows the user is most interested in analyzing. The interface offers subsetting capabilities, which allows a user to dynamically choose a traffic volume and sort the flows (ascending or descending) to obtain top-k (one can also vary k) flows in an order preferential to the user using sliders. Users can search for a particular SD pairs or a traffic volume to locate the flow. In Figure 3, one can observe the top-5 SD flows filtered by the flow volumes (306-5725) and SD Pair ids (1298-3501) sorted in ascending order.

Source Destination Subsetting - We allow a user to closely analyze network flow over a small subset of the generated network. The flow values that we obtain from the algorithm are represented as a scatter plot for analysis. Once selected, a subgraph (of selected flows) is obtained, and users can interactively visualize the effect of a change in parameters (prior, path, change in route etc.) on flow values.

Visualizing $\mathcal{A}\mathcal{A}^T$ - Database optimization for SRP is based on the observation that, bulk of the values of $\mathcal{A}\mathcal{A}^T$ is concentrated in a small number of values(along the major diagonal). We allow the user to interactively view the effect of using a threshold on $\mathcal{A}$ and $\mathcal{A}\mathcal{A}^T$ for its processing. The heat map on the left panel of Figure 4 shows the sparseness of the $\mathcal{A}\mathcal{A}^T$ matrix as well as the domination of the diagonal values over the corresponding row and column. Far fewer non zero values on right panel shows effect of thresholding in computation of $\mathcal{A}\mathcal{A}^T$. We highlighted the same cell (row-4, column-37) in both panels to show the optimization through thresholding (left calculated as 137 and right was never calculated, as 137 is lesser than the threshold, 455). One can clearly notice the pixel density(as a gradient of value associated with it) of the cell to turning from a light shade(on left) of white to complete dark (on right).

3. DEMONSTRATION PLAN

This section details the implementation details of Orca-SR, it's demonstration plan and comparison portfolios.

3.1 Demonstration scenarios

Datasets. We experimented Orca-SR with all datasets mentioned in the works by Asudeh *et.al.* [2] and reproduced the results. For this demonstration, we highlight three different datasets – synthetic Erdős-Rényi graph N_1 and two real network datasets $p2p_1$ and $p2p_2$ from SNAP[5] repository.

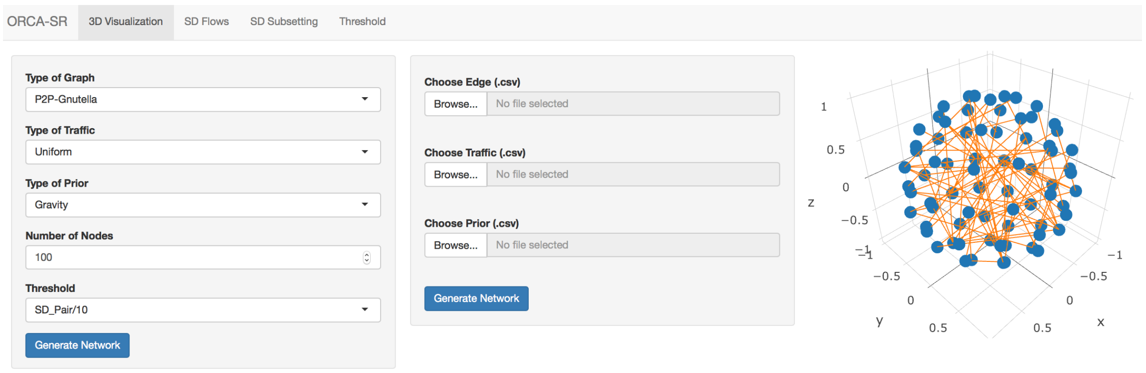
User Interaction Model. On load, the web app introduces a user to network, traffic and threshold selection page. A user can choose a dataset (one of N_1, $p2p_1$ and $p2p_2$), traffic generation model (*Pareto [5]*, *normal*, *exponential*), choice of traffic prior (*preferred: gravity [9]*, *normal*, *random*). The prior is a critical measure as our final solution is a point in the solution space that is closer to the chosen prior. As the final input, a user may provide the choice of the *threshold*(*preferred: fractions of 10 on # of Nodes*) value, the one which determines the accuracy of the results and computational time for the results. Besides, users can provide a new dataset to the system as *.csv* file.

Active Real-Time SRP. While prior state-of-the-art methods computed the traffic matrix in a couple of hours, Orca-SR completes the SD pairs traffic estimation in a few minutes, demonstrating orders of magnitude improvement. We demonstrate the performance improvements based on large real-world networks. Traffic volume generated by SD pairs change instantaneously and previously, network administrator had to wait for a couple of hours for re-estimation with changes in b. With Orca-SR and its optimizations, we demonstrate to track these changes and reflect the changes in SD pairs traffic over the network in real-time. Users can select the frequency of change in traffic from the drop-down menu in the *visualization* tab and the system automatically updates the traffic instantaneously.

Dynamic Network Adaptation. Real networks evolve dynamically and a network administrator would need to get a macroscopic view of the network adapting to these changes for planning resources. Often changes in the network requires a complete re-computation in previous solution frameworks and were impractical for real-time system monitoring. Orca-SR can dynamically capture these network changes and adapt without recomputing the solution through selective reuse of signature matrix [2]. Users can choose to update the network from the *SD Flows* tab with frequency of addition(deletion) or the number of new(deleted) nodes. Orca-SR dynamically adapts and adjusts to these network changes in real-time.

[4]https://en.wikipedia.org/wiki/Erd%C5%91s%E2%80%93R%C3%A9nyi_model

[5]https://snap.stanford.edu/



Type of Graph: P2P-Gnutella

Type of Traffic: Uniform

Type of Prior: Gravity

Number of Nodes: 100

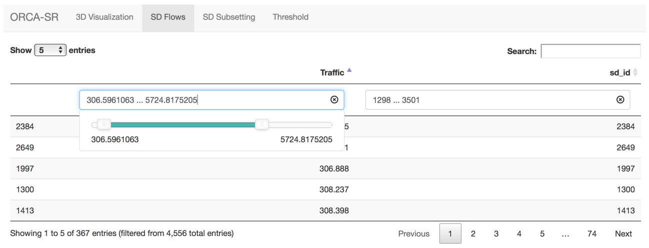
Threshold: SD_Pair/10

Choose Edge (.csv): No file selected

Choose Traffic (.csv): No file selected

Choose Prior (.csv): No file selected

Figure 2: Orca-SR Interactive Network Simulation and Visualization interface



Show 5 entries

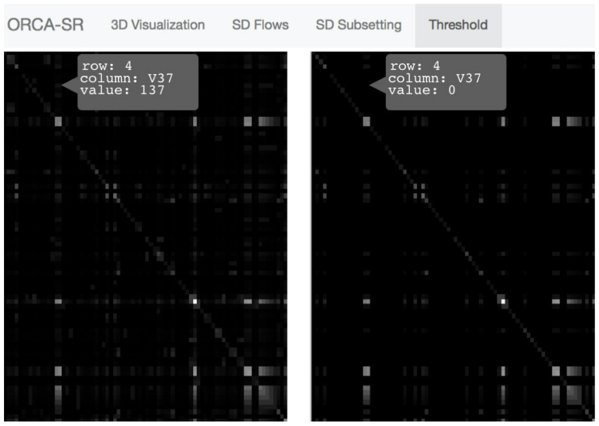
Search:

	Traffic	sd_id
	306.5961063 ... 5724.8175205	1298 ... 3501
2384	5	2384
2649	1	2649
1997	306.888	1997
1300	308.237	1300
1413	308.398	1413

306.5961063 5724.8175205

Showing 1 to 5 of 367 entries (filtered from 4,556 total entries)

Figure 3: Interactive selection of top-k SD Flows



```
row: 4
column: V37
value: 137
```

```
row: 4
column: V37
value: 0
```

Figure 4: Visualizing $\mathcal{A}\mathcal{A}^T$ without and with thresholding

Top-k Flows Detection and Real-Time Network Subsetting: Network administers are also interested in microscopic view of the network for resource allocation including the top-k flows. ORCA-SR enables an administrator to identify the top flows [2] and further facilitates a user to focus on a subset of the network with top-k flows for analysis. In this demonstration, once a sub-network is chosen the system automatically adapts to the sub-network, allowing a user to update traffic flows in the sub-network.

4. ACKNOWLEDGMENTS

The work of Gautam Das was supported in part by grants 1745925 and 1937143 from the National Science Foundation, and a grant from AT&T. We thank Balan Ramesh for his insightful discussions on developing the shiny R web application.

5. REFERENCES

[1] A. Asudeh, J. Augustine, A. Nazi, S. Thirumuruganathan, N. Zhang, G. Das, and D. Srivastava. Efficient signal reconstruction for a broad range of applications. *SIGMOD Records, Special Issue on 2018 ACM SIGMOD Research Highlights*, 48(1):42–49, Nov. 2019.

[2] A. Asudeh, J. Augustine, A. Nazi, S. Thirumuruganathan, N. Zhang, G. Das, and D. Srivastava. Scalable algorithms for signal reconstruction by leveraging similarity joins. *The VLDB Journal, Special Issue on best of VLDB'18*, pages 1–27, 2019.

[3] A. Asudeh, A. Nazi, J. Augustine, S. Thirumuruganathan, N. Zhang, G. Das, and D. Srivastava. Leveraging similarity joins for signal reconstruction. *PVLDB*, 11(10):1276–1288, 2018.

[4] S. Chaudhuri, V. Ganti, and R. Kaushik. A primitive operator for similarity joins in data cleaning. In *ICDE*. IEEE, 2006.

[5] J. Gordon. Pareto process as a model of self-similar packet traffic. In *Global Telecommunications Conference, 1995. GLOBECOM'95., IEEE*, volume 3, pages 2232–2236. IEEE, 1995.

[6] M. Hadjieleftheriou, X. Yu, N. Koudas, and D. Srivastava. Hashed samples: selectivity estimators for set similarity selection queries. *PVLDB*, 1(1):201–212, 2008.

[7] L. Kleinrock and F. Kamoun. Hierarchical routing for large networks performance evaluation and optimization. *Computer Networks*, 1(3):155–174, 1977.

[8] P. Tune and M. Roughan. Maximum entropy traffic matrix synthesis. *ACM SIGMETRICS Performance Evaluation Review*, 42(2):43–45, 2014.

[9] Y. Zhang, M. Roughan, N. Duffield, and A. Greenberg. Fast accurate computation of large-scale ip traffic matrices from link loads. In *SIGMETRICS*, volume 31, pages 206–217. ACM, 2003.