Efficient Signal Reconstruction for a Broad Range of Applications

Abolfazl Asudeh[†], Jees Augustine[‡], Azade Nazi[§], Saravanan Thirumuruganathan[¶], Nan Zhang[‖], Gautam Das[‡], Divesh Srivastava[**]
[†] University of Illinois at Chicago; [‡] University of Texas at Arlington; [§] Google Brain
[¶]QCRI, HBKU; [‖] Pennsylvania State University; [**]AT&T Labs-Research
asudeh@uic.edu, {jees.augustine@mavs, gdas@cse}.uta.edu,
azade.nazi@google.com, sthirumuruganathan@hbku.edu.qa, nan@ist.psu.edu,
divesh@research.att.com

ABSTRACT

The signal reconstruction problem (SRP) is an important optimization problem where the objective is to identify a solution to an under-determined system of linear equations $AX = b$ that is closest to a given prior. It has a substantial number of applications in diverse areas including network traffic engineering, medical image reconstruction, acoustics, astronomy and many more. Most common approaches for solving SRP do not scale to large problem sizes. In this paper, we propose a dual formulation of this problem and show how adapting database techniques developed for scalable similarity joins provides a significant speedup when the A matrix is sparse and binary. Extensive experiments on real-world and synthetic data show that our approach produces a significant speedup of up to 20x over competing approaches.

1. INTRODUCTION

The database community has been at the forefront of grappling with challenges of big data and has developed numerous techniques for the scalable processing and analysis of massive datasets. These techniques often originate from solving core data management challenges but then find their way into effectively addressing the needs of big data analytics. For example, efficiency of machine learning has been successfully boosted by database techniques as varied as materialization, join optimization, query rewriting for efficiency, query progress estimation, federated databases, etc. This paper studies how database techniques can benefit another foundational problem in big data analytics, *large-scale signal reconstruction* [22], which is of significant interest to research communities such as computer networks [25], medical imaging [11, 14], etc. We demonstrate that the scalability of existing solutions can be significantly improved using ideas originally developed for similarity joins [7] and selectivity estimation for set similarity queries [3, 12].

Signal Reconstruction Problem (SRP): The essence of SRP is to solve a linear system of the form $AX = b$, where X is a high-dimensional unknown *signal* (represented by an m-d vector in $\mathbb{R}^m$), b is a low-dimensional projection of X that can be observed in practice (represented by an n-d vector in $\mathbb{R}^n$ with $n \ll m$), and A is a $n \times m$ matrix that captures the linear relationship between X and b. There are many real-world applications that follow the SRP model (see § 2.1). For example, high-dimensional signals like environmental temperature can only be observed through low-dimensional observations, like readings captured by a small number of temperature sensors. Similarly, as further explained in § 2.1, end-to-end network traffic, another high-dimensional signal, is often monitored through low-dimensional readings such as traffic volume on routers in the backbone or edge networks. In these applications, the laws of physics or the topology of computer networks reveal the value of A, and our objective is to reconstruct the high-dimensional signal X from the observation b based on the knowledge of A.

Since $n \ll m$, the linear system is underdetermined. That is, for a given A and b, there are an infinite number of feasible solutions (of X) that satisfy $AX = b$ [13, 22]. In order to identify the best reconstruction of the signal, it is customary to define and optimize for a *loss function* that measures the distance between the reconstructed X and a prior understanding of certain properties of X. For example, one can represent one's prior belief of X as an m-d vector X', and define the loss function as the ℓ_2-norm of $X - X'$, i.e., $\|X - X'\|_2$. In other cases, when prior knowledge indicates that X is sparse, one can define the loss function as the ℓ_0-norm of X, aiming to minimize the number of non-zero elements in the reconstructed signal. For the purpose of this paper, we consider the ℓ_2-based loss function of $\|X - X'\|_2$, which has been adopted in many application-oriented studies such as [11, 25].

Running Example of SRP: While SRP has a broad range of applications, for the ease of discussion, it is important to have a running example of SRP on a domain-specific application. What we use as a running example of SRP throughout the paper is a common instance of network tomography (§ 2.1.1), where the objective is to compute the pairwise end-to-end traffic in IP Networks. Pairwise traffic measures the volume of traffic between all pairs of source-destination nodes in an IP network, and has numerous uses such as capacity planning, traffic engineering and detecting traffic anomalies. Informally, consider an IP network where various sources and destinations send different amounts of traffic to each other. The network administrator is aware of the network topology and the routing table (from which we can construct matrix A). In addition, the administrator can observe the traffic passing through each link in the backbone network (observation b). The goal is to find the amount of traffic flow between all source-destination pairs (signal X). Note that one cannot directly measure the raw traffic between all source-destination pairs due to challenges in instrumentation and storage - see [25, 26] for a technical discussion. In almost all real-world IP networks, the number of source-destination pairs is significantly larger than the number of links, leading to an underdetermined linear system. To reconstruct the pairwise traffic, the network community introduced various traffic models, e.g., the

DOI: https://doi.org/10.14778/3231751.3231752

gravity model [25], as the prior for X', and used the ℓ_2-distance between X and the prior as the loss function. Note that in reconstructing the pairwise distances, efficiency is a concern front-and-center, especially given the rise of Software Designed Networks (SDNs) which feature much larger sizes and much more frequent topological changes, pushing further the scalability requirements of signal reconstruction algorithms.

Research Gap: Because of the importance of SRP, there has been extensive work from multiple communities on finding efficient solutions. To solve the problem efficiently, methods explored in the recent literature include statistical likelihood based iterative algorithms based on expectation-maximization [5], as well as the use of linear algebraic techniques such as computing the pseudoinverse of A [22] or performing Singular Value Decomposition (SVD) on A, and iterative algorithms for solving the linear system [22]. Yet even these approaches cannot scale to fully meet the requirements in practice, especially the traffic reconstruction needs of large-scale IP networks - which call for a more scalable solution [26].

Our Approach: In this paper, we consider a special case of SRP where A, X, b are non-negative with A being a *sparse binary matrix*. Such a setting finds its applications in many domains, as explained in § 2.1.

Our proposed solution starts with an exact algorithm based on the transformation of the problem into its Lagrangian dual representation. As we shall show in § 6, our algorithm DIRECT, which directly computes X through the dual representation, already outperforms commonly used approaches for SRP, as it avoids expensive linear algebraic operations required by the previous solutions. Next, we investigate whether our approach can be sped up even further, by replacing exact computations with approximation techniques. This can be useful in applications where the user is willing to trade accuracy for efficiency. We carefully investigate the computational bottlenecks of DIRECT and find it to be a special case of matrix multiplication involving a sparse binary matrix with its transpose. We start by investigating a seemingly straightforward sampling strategy for approximately computing this matrix multiplication, but encounter a negative result. Then, we use the observation that a small number of cells in the result matrix of the bottleneck operation take the bulk of the values, and propose a threshold-based algorithm for approximating it. Specifically, we reduce the problem to computing the dot product of two vectors if and only if their similarity is above a user-provided threshold. Our key idea here is to leverage various database techniques to speed up the multiplication operation. We propose a hybrid algorithm based on a number of techniques originally proposed for computing similarity joins and selectivity estimation of set similarity queries, resulting in significant speedup in solving SRP in comparison with the exact solution.

Experimental Summary: We conduct extensive experiments on both real-world and synthetic datasets with a special emphasis on traffic matrix computation. We compare our method against a number of commonly used approaches such as an efficient quadratic programming based solver, a two stage approximate approach first proposed in [25] and one based on compressive sensing. Our experimental results show that our exact algorithm significantly outperforms the baselines. Furthermore, our threshold based approximation approaches inspired from similarity joins provide even more speedup over DIRECT without resulting in any significant increase in reconstruction error.

Summary of Contributions:

- We investigate the Signal Reconstruction Problem (SRP) which has diverse applications. By using techniques that were originally pioneered for databases, we dramatically improve the scale of problems that could be solved.
- We formulate SRP as a Quadratic Programming problem and derive its Lagrangian dual form and propose an exact algorithm DIRECT to solve the dual problem. Our algorithm DIRECT already outperforms commonly used approaches for SRP.
- We identify the computational bottleneck in DIRECT and propose a threshold-based algorithm for approximating it. We propose a hybrid algorithm that combines two algorithms that were designed for efficiently computing set similarity joins.
- We conduct a comprehensive set of experiments on both real and synthetic datasets that confirm the efficiency and effectiveness of our approach, and report the results in [1]. Here we provide a summary of those results.

Paper Organization: We provide the necessary background to SRP and formally define it in § 2. In § 3, we describe the exact algorithm DIRECT for solving SRP. In § 4, we show how to apply approximation using techniques from databases to significantly speed up the computation. In § 5, we discuss how our approach can be easily adapted to identify the top-K components of the reconstructed signal. § 6 describes our experiments followed by related work in § 7 with § 8 providing the conclusion.

2. PROBLEM FORMULATION

As mentioned in Section 1, we consider a special class of SRP that has a number of applications in network traffic engineering, tomographic image reconstruction and many others, discussed in § 2.1. We are given a system of linear equations $AX = b$ where

- $A \in \{1, 0\}^{n \times m}$ is a sparse binary matrix $n \ll m$.
- $X \in \mathbb{R}^m$ is the "signal" to be reconstructed and is a vector of unknown values.
- $b \in \mathbb{R}^n$ is the vector of observations.

Each row in the matrix A corresponds to an equation with each column corresponding to an unknown variable. When the number of equations (n) is much smaller than the number of unknowns (m), the system of linear equations is said to be under-determined and does not have a unique solution. The solution space can be represented as a hyperplane in a $m' \in [2, m]$ dimensional vector space[1]. Since SRP does not have a unique solution, one must have auxiliary criteria to choose the best solution from the set of (possibly infinite) valid solutions. A common approach in SRP is to provide a prior X' and the objective is to pick the solution X that is closest to X'. We study the problem where the objective is to find the point satisfying $AX = b$ that minimizes the ℓ_2 distance from a prior point X'. Formally the problem is defined as:

$$\begin{aligned} &\min \ \|X - X'\|_2 \\ &\text{s.t. } AX = b \end{aligned} \tag{1}$$

Figure 1 provides an example visualization of the problem in 3 dimensions. The gray plane is the solution space with the prior marked as a point X'. The intersection of the perpendicular line to the plane that passes though X' is the point that minimizes $\|X - X'\|_2$.

In this paper, we pay attention to the fact that SRP is a special case of quadratic programming where (a) the constraints are only in the form of equality, (b) matrix A is sparse, and (c) matrix A is binary (and hence un-weighted). By leveraging these characteristics, we seek to design more efficient solutions compared with the baselines that are designed for general cases. Especially, in § 3, after studying the existing work, we use the dual representative of the

[1]We assume that the problem has at least one solution.

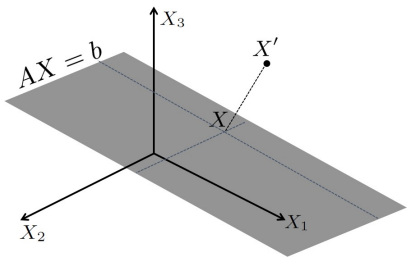

Figure 1: Visualizing the problem

problem to propose an efficient exact algorithm. Later in § 4, we show how leveraging similarity join techniques help in achieving significant speed up without sacrificing much accuracy.

2.1 Broad Application Range

SRP covers a broad range of real world problems that use signal reconstruction. In practice, it is popular to observe low-dimensional projections in form of (unweighted) aggregates of a high-dimensional signal vector. For example, in general network flow applications (such as road traffic estimation [27]), the value on each edge is the summation of the flows values which includes this edge as part of the path between them. Of course, a requirement to our problem is an "expert-provided" prior template, such as *gravity model* [25] for the network flow problems. Another major application domain for SRP problem over aggregates is image reconstruction (§ 2.1.2), where observations are unweighted projections of unknowns. Image reconstruction has a broad applications ranging from medical imaging [11, 14], to astronomy [23] and physics [19] and to gas flow reconstruction [2]. Some of the other applications of SRP, in general, include radar data reconstruction [18], transmission electron microscopy [15], and product assortment design [9], to name a few. To showcase some applications in more detail, we sketch a few examples in the context of network flow problems and image reconstruction in the following.

2.1.1 Network Tomography

Traffic matrix computation *(the running example)*: Consider an IP network with n traffic links and m source-destination traffic flows (SD flow) between the ingress/egress points, where $n \ll m$. The ingress/egress points can be PoPs (points of presence) or routers or even IP prefixes depending on the level of granularity required. The network has a routing policy prescribes a path for each of the SD flows that can be captured in a $\#links(n) \times \#flows(m)$ binary matrix A, where the entry $A[i, j] = 1$ if the link i is used to route the traffic of the j-th SD flow. The matrix A is sparse and "fat" with more SD flows(columns) than number of links(rows). Note that, one cannot directly measure each of the SD flows on a link owing to efficiency reasons. However, one can easily measure the total volume of the network traffic that passes through a given link using network protocols such as SNMP. Thus, the load on each link i becomes the observed vector b. To obtain a prior X', one can use any traffic model such as the popular and intuitive *gravity model* [25]. It assumes independence between source and destination and states that traffic between any given source s and destination d is proportional to the product of network traffic entering at s and that exiting at d.

Traffic analysis attack in P2P networks: In traffic analysis attack, the information leak on traffic data is exploited to expose the user traffic pattern in P2P networks [10]. Here we propose the following traffic analysis attack that can be modeled to our problem: Consider an adversary who monitors the link level traffics in a P2P network. Applying SRP, one can directly identify the volume of traffic between any pair of users in a P2P network.

2.1.2 Image Reconstruction

Image reconstruction [16,24] has a wide range of applications in different fields such as medical imaging [11, 14], and physics [19]. Given a set of (usually 2D) projection of a (usually 3D) image, the objective is to reconstruct it. The reconstruction is usually done with the help of some prior knowledge. For example, knowing that the 2D projections are taken from a human face, one may use a template 3D face photo and, among all possible 3D reconstructions from the 2D images, find the one that is the closest to the template, making the image reconstruction more effective.

CT Scan: A popular application of SRP is tomographic reconstruction, which is a multi-dimensional linear inverse problem with wide range of applications in medical imaging [11, 14] such as CT scans (computed tomography). Informally, a CT scan takes multiple 2D projections (b) through X-rays from different angles (A) and the objective is to reconstruct the 3D image from the projections. Note that many 3D images may produce the same projections necessitating the use of priors to choose an appropriate reconstruction.

Radio astronomy: In Astronomy, SRP has application for reconstructing interferometric images where the astrophysical signals are probed through Fourier measurements. The objective is to reconstruct the images from the observations – forming a SRP scenario. Also, the specific prior information about the signals plays an important role in reconstruction, as mentioned in [23].

3. EXACT SOLUTION FOR SOLVING SRP

In this section, we begin by describing two representative approaches for solving SRP from prior research and highlight their shortcomings. We then propose a dual representation of the problem that can be solved exactly in an efficient manner and already outperforms the baselines. This alternate formulation has a number of appealing properties that allows one to leverage various database techniques for speeding it up.

3.1 Lagrangian Formulation of SRP

In this subsection, we leverage the Lagrangian dual form of SRP as a special case of quadratic programming, and design an efficient exact solution for it. For SRP as specified in Equation 1, $f(X) = \frac{1}{2}X^TX - X'^TX$ and $g(X) = AX$.[2] Thus, our problem can be re-written as:

$$L(X, \lambda) = \frac{1}{2}X^TX - X'^TX + \lambda^T(AX - b) \tag{2}$$

Next, we find the stationary point [3] of Equation 2 in the general form by taking the derivatives with regard to X and λ and setting them to zero, we get:

$$X = X' - A^T(AA^T)^{-1}(AX' - b) \tag{3}$$

Solving SRP in Dual Form. The stationary point of Equation 2 is the optimal solution for our problem (Equation 1). In contrast to prior work, we solve the SRP problem by directly solving Equation 3. We make two observations. First, the matrix $AA^T \in \mathbb{Z}^{n \times n}$ always has an inverse as it is full-rank. From Figure 1, one can note that the problem has a unique solution that minimizes the distance from the prior. It means that AA^T is full-rank, because otherwise the problem was not feasible and would not have a solution. Second, Equation 3 does have a matrix inverse operator that is expensive to compute. However, one can avoid taking the inverse of AA^T

[2]Note that $\min \frac{1}{2}X^TX - X'^TX$ is the same as $\min ||X - X'||_2$.

[3]Since, looking at Figure 1, Equation 1 has a single optimal point, Equation 2 has one stationary point which happens to be the saddle point.

0	0	0	1	0	0	0	1	0	0
0	0	1	0	0	0	0	0	0	0
0	0	0	0	0	1	0	1	0	1
0	1	0	0	0	0	1	0	0	0

(a)

$\langle 3, 7 \rangle$
$\langle 2 \rangle$
$\langle 5, 7, 9 \rangle$
$\langle 1, 6 \rangle$

(b)

Figure 2: Illustration of the sparse representation of A. (a) Non sparse representation, (b) Sparse representation

by computing ξ in Equation 4, and replacing $(AA^T)^{-1}(AX' - b)$ by it in Equation 3.

$$(AA^T)\xi = AX' - b \tag{4}$$

Algorithm 1 provides the pseudocode for DIRECT.

Algorithm 1 DIRECT
Input: A, b, and X'
Output: X

1: $t = AA^T$
2: $t_2 = AX' - b$
3: Solve system of linear equations: $t\,\xi = t_2$
4: $X = X' - A^T\xi$
5: **return** X

Performance Analysis of DIRECT**.** Let us now investigate the performance of our algorithm. Recall that A is a fat matrix with $n \ll m$ while X and X' are m-dimensional vectors, and b is a n-dimensional vector. Line 1 of Algorithm 1 takes $O(n^2m)$ while Line 2 takes $O(nm)$. Line 3 involves solving a system of linear equations. A naive way would be to compute the inverse of t that can take as much as $O(n^3)$. However, by observing that t is sparse, one can use approaches such as Gauss-Jordan elimination or other iterative methods that are practically much faster for sparse matrices. Finally, the computation of Line 4 is in $O(nm)$. Looking at DIRECT holistically, one can notice that its computational bottleneck is Line 1 thereby making the overall complexity to be $O(n^2m)$.

An additional approach to speedup DIRECT is to observe that matrix A is sparse and thereby store it in a manner that allows efficient matrix multiplication. Since A is binary (and hence unweighted), a natural representation is to store only the indices of non-zero values. Figures 2a and 2b show the non-sparse and sparse representation of a matrix A. Note that AA^T is symmetric since $t[i, j]$ and $t[j, i]$ are obtained by the dot product of rows i and j of A. Let l be the number of non-zero elements in each row. Since A is sparse, $l \ll m$, one can design a natural matrix multiplication algorithm with time complexity of $O(nml)$ that is orders of magnitude faster than algorithm such as Strassen algorithm.

4. TRADING OFF ACCURACY WITH EFFICIENCY

In many applications of SRP, m is often in $O(n^2)$, thereby making the computational complexity of DIRECT to be $O(n^4)$. The key bottleneck is the computation of AA^T. On the other hand, for large problem instances, the user may accept trading off accuracy with efficiency and prefer a close-to-exact solution that is computed quickly, rather than the expensive exact solution. In this section, our objective is to speed up DIRECT by computing the bottle-neck step, i.e., computing AA^T, approximately. We show how to leverage a threshold-based approach by only computing the values of matrix AA^T that are larger than a certain threshold. We describe the connection between this problem variant and similarity joins and propose a hybrid method by adopting two classical algorithms designed for similarity estimation, which results in an efficient solution for computing AA^T.

4.1 Bounding Values in Matrix AA^T

We begin by showing that one can efficiently compute the bound for each cell value in matrix AA^T. Figure 3 shows a sparse matrix A with 183 rows and 495 columns, in which the non-zero elements are highlighted in white. Figure 4 shows the non-zero elements in matrix AA^T. We can notice that AA^T is square and also sparse due to the fact that every element of AA^T is the dot product of two sparse vectors (two rows of matrix A). Furthermore, one can also observe a more subtle phenomenon that we state in Theorem 1 which could used to design an efficient algorithm.

THEOREM 1. *Given a sparse binary matrix A, considering the elements on the diagonal of AA^T, i.e., $t[i, i]$, $\forall 0 \leq i < n$:*

- *$t[i, i] = |A[i]|$, where $|A[i]|$ is the number of non-zero elements in row $A[i]$.*
- *$t[i, i]$ is an upper bound for the elements in the row $t[i]$ and the column $t[, i]$; formally, $\forall 0 \leq j < n : t[i, j] \leq t[i, i]$ and $t[i, j] \leq t[j, j]$.*

The proof can be found in [1].

Consider two representations of AA^T of the example matrix given in Figure 3. Figure 4 shows all the non-zero elements of AA^T while Figure 5 shows a magnitude-weighted variant wherein cells with larger values are plotted in brighter colors. Figure 5 visually shows that the elements on the diagonal are brighter than the ones in the same row and column as predicted by Theorem 1. Furthermore, one may notice that most of the non-zero elements of AA^T (in Figure 4) are small values (in Figure 5). This shows that while there are a reasonable number of non-zero elements, the number of elements with higher magnitude is often much smaller. Next, we use this insight along with Theorem 1 for speeding up DIRECT.

4.2 Threshold Based Computation Of AA^T

In the previous subsection, we discussed the bound on the cell values in AA^T and showed that a small number of elements in AA^T take the bulk of the value. This is the key in designing a threshold-based algorithm for computing AA^T wherein we only compute values of AA^T that are above a certain threshold. Specifically, we use the elements on the diagonal as an upper-bound and only compute the elements for which this upper-bound is larger than a user-specified threshold. Note that, if the threshold is equal to 1, the algorithm will compute the values of all elements. However, the user-specified threshold allows additional opportunities for efficiency.

Algorithm 2 provides the pseudocode for the threshold-based multiplication of sparse binary matrix A with its transpose. This algorithm depends on the existence of an oracle called SIM that given two rows $A[i]$ and $A[j]$, and the threshold τ, returns the dot product of $A[i]$ and $A[j]$ if the result is not less than τ.

4.3 Leveraging Similarity Joins for Oracle SIM

The database community has extensively studied mechanisms for computing set similarity for applications such as data cleaning [7] where the objective is to efficiently identify the set of tuples that are "close enough" on multiple attributes. In this subsection, we describe how to implement the oracle SIM by leveraging prior research on computing set similarity. Especially, we propose a hybrid method that combines the threshold-based similarity joins with the sketch-based methods to resolve their shortcomings.

Oracle SIM through Set Similarity. Given two rows $A[i]$ and $A[j]$, and the threshold τ, SIM should find the dot product of $A[i]$

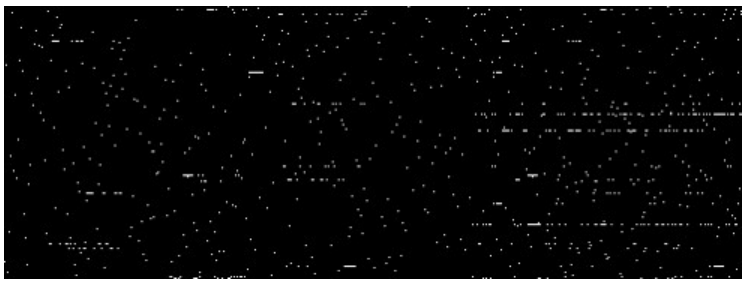

Figure 3: An example of the binary sparse matrix $A_{183\times495}$

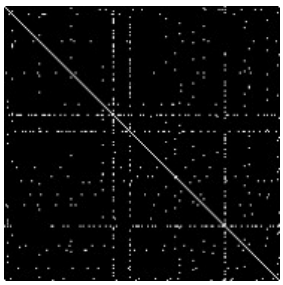

Figure 4: The non-zero elements in AA^T for the example of Figure 3

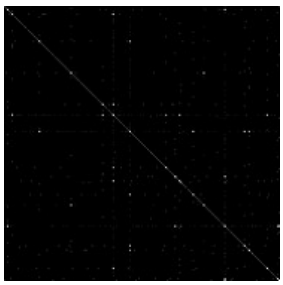

Figure 5: Magnitude of weights in AA^T for the example of Figure 3

Algorithm 2 ApproxAA^T
Input: Sparse matrix A, Threshold τ
Output: t

```
1: F = {}
2: for i = 0 to n − 1 do
3:     t[i, i] = |A[i]|
4:     if |A[i]| ≥ τ then add i to F
5: end for
6: for every pair i, j ∈ F do
7:     t[i, j] = t[j, i] = SIM(A[i], A[j], τ)
8: end for
9: return t
```

and $A[j]$ if it is not less than τ. It is possible to make an interesting connection between SIM and sets similarity problems as follows. Let every column in matrix A be an object o in a universe $\mathcal{U}$ of m elements. Every row $A[i]$ represents a set U_i in $\mathcal{U}$, where $\forall o_j \in \mathcal{U}$, $o_j \in U_i$ *iff* $A[i,j] = 1$. Equivalently, each row corresponds to a set U_i that stores the indices of the non-zero columns similar to Figure 2b. Using this transformation, we can see that our objective is to compute $|U_i \cap U_j|$ for all pairs of sets U_i and U_j where $|U_i \cap U_j| \geq \tau$. Note that we represent $|U_i \cap U_j|$ by $\cap_{i,j}$ and $|U_i \cup U_j|$ by $\cup_{i,j}$ respectively.

Due to its widespread importance, different versions of this problem have been extensively studied in the DB community. In this paper, we consider one exact approach and two approximate approaches based on threshold-based algorithms [7] and sketch-based methods [3, 8, 12]. We then compare and contrast the two approximate approaches, describe the scenarios when they provide better performance, and propose a hybrid algorithm based on these scenarios.

Exact Approach : Set Intersection. One can see that when $\tau = 1$, the problem boils down to computing AA^T exactly. This in turn, boils down to computing the intersection between two sets as efficiently as possible. The sparse representation of the matrix often provides the non-zero columns in an ordered manner. The simplest approaches for finding the intersection of ordered sets is to perform a linear merge by scanning both the lists in parallel and leveraging the ordered nature similar to the merge step of merge-sort. One can also speedup this approach by using sophisticated approaches such as binary search on one of the lists or using sophisticated data structures such as treaps or skip-lists. Each of these approaches allows one to "skip" some elements of a set when necessary.

Approximate Approach : Threshold based Algorithms. Threshold-based algorithms, such as [7] identify the pair of sets such that their similarity is more than a given threshold. This has a number of applications such as data cleaning, deduplication, collaborative filtering, and product recommendation in advertisement where the objective is to quickly identify the pairs that are highly similar. The key idea is that if the intersection of two sets is large, the intersection of small subsets of them is non zero [7]. More precisely, for two sets U_i and U_j with size h, if $\cap_{i,j} \geq \tau$, any subsets $U_i' \subset U_i$ and $U_j' \subset U_j$ of size $h - \tau + 1$ will overlap; i.e., $|U_i' \cap U_j'| > 0$. Using this idea, while considering an ordering of the objects, the algorithm first finds the set of candidate pairs that overlap in a subset of size $h - \tau + 1$. In the second step, the algorithm verifies the pairs, by removing the false positives.

One can see the effectiveness of this method highly depends on the value of τ and, considering the target application, it works well for the cases that τ is large. For example, consider a case where $h = 100$. When $\tau = 99$ (i.e., 99% similarity), the first filtering step needs to compare the subsets of size 2 and is efficient; whereas if $\tau = 10$, the filtering step needs to compare the subset pairs of size 91, which is close to the entire set. The latrer case is quite possible in our problem. To understand it better, let us consider matrix A in Figure 3, while setting τ equal to 5 in Algorithm 2. Even though the size of many of the rows is close to the threshold, there are rows $A[i]$ where $|A[i]|$ is significantly larger than it. For example, for two rows $A[i]$ and $A[j]$ where $|A[i]| \geq 50$ and $|A[j]| \geq 50$, to satisfy the dot product be not less than τ, the filtering step needs to compare the subsets of size ≥ 44, which is close to the exact comparison of $A[i]$ and $A[j]$.

Approximate Approach : Sketch based Algorithms. Sketch based methods such as [3, 8, 12] use a precomputed synopsis such as a minhash for answering different set aggregates such as Jaccard similarity. The main idea behind the min-hashing [4] based algorithms is as follows: consider a hash (ordering) of the elements in $\mathcal{U}$. For each set U_i, let $h_{\min}(U_i)$ be the element $o \in U_i$ that has the minimum hash value. Two sets U_i and U_j have the same min-hash, when the element with the smallest hash value belongs to their intersection. Hence, it is easy to see that the probability that $h_{\min}(U_i) = h_{\min}(U_j)$ is equal to $\frac{\cap_{i,j}}{\cup_{i,j}}$, i.e., Jaccard similarity of U_i and U_j. Bottom-k sketch [8], a variant of min-hashing picks the hash of the k elements in U_i with the smallest hash value, as its signature. The Jaccard similarity of two sets U_i and U_j is estimated as $\frac{k_\cap(i,j)}{k}$, where $k_\cap(i,j)$ is $|h_k(U_i) \cap h_k(U_j)|$. Bayer et al. [3] use the bottom-k sketch for estimating the union and intersection of the sets. Let $h_{i,j}[k]$ be the hash value of the k-th smallest hash value in $h_k(U_i) \cup h_k(U_j)$. The idea is that the larger the size of a set is, the smaller the expected value of the k-th element in hash is. Using the results of [3], $\frac{m(k-1)}{h_{i,j}[k]}$ is an unbiased estimator for $\cup_{i,j}$. Hence the estimation for $\cap_{i,j}$ is as provided in Equation 5.

$$E[\cap_{i,j}] = \frac{k_\cap(i,j)}{k} \frac{m(k-1)}{h_{i,j}[k]} \tag{5}$$

Estimating $\cup_{i,j}$ with Equation 5, performs well when $\cup_{i,j} \gg$

1 [3], i.e., the larger sets. Hence, we combine the threshold-based and sketch-based algorithms to design the oracle SIM, as a hybrid method that, based on the sizes of the rows $A[i]$ and $A[j]$, adopts the threshold-based computation with sketch-based estimation for computing the dot product of $A[i]$ and $A[j]$. We consider $\log(m)$ as the threshold to decide which strategy to adopt. Considering the effectiveness of threshold based approaches when U_i and U_j are small and, as a result, the two sets need a large overlap to have the intersection larger than τ, if $|U_i|$ and $|U_j|$ are less than $\log(m)$, we choose the threshold-based intersection computation. However, if the size of U_i or U_j is more then we use the bottom-k sketch, while considering k to be $\log(m)$. For each element $o_j \in \mathbb{U}$, we set $h(o_j) = j$. Hence, for each vector U_i the index of the first $\log(m)$ elements in it are its bottom-k sketch. Using this strategy, Algorithm 3 shows the pseudo code of the oracle SIM.

Given two given sets U_i and U_j (corresponding to the rows $A[i]$ and $A[j]$) together with the threshold τ, the algorithm aims to compute the value of $\cap_{i,j}$, if it is larger than τ. Combining the two aforementioned methods, if $|U_i|$ and $|U_j|$ are more than a value α, the algorithm uses sampling to estimate $\cap_{i,j}$, otherwise it applies the threshold-based method to compute it. During the sampling, rather than sampling from $\mathcal{U}$, the algorithm samples from U_i to reduce the underestimation of probability. In this case, in order to compute $\cap_{i,j}$, the algorithm, for each sample, picks a random object from U_i and check its existence in U_j. It is easy to see it is an unbiased estimator for $\cap_{i,j}$, where its expected value is $\cap_{i,j}$. If $|U_i|$ or $|U_j|$ is less than α, the algorithms applies threshold-based strategy for computing $\cap_{i,j}$. As discussed earlier in this subsection, in order for $\cap_{i,j}$ to be more than τ, the subsets of size $\cap_{i,j} - \tau + 1$ should intersect. Hence, the algorithm first applies the threshold filtering and only if the two subsets intersect it continues with computing $\cap_{i,j}$.

Algorithm 3 SIM
Input: the sets U_i and U_j, Threshold τ
Output: c

if $|U_i| \geq \log(m)$ and $|U_j| \geq \log(m)$ **then**
  $h_i =$ the first k elements in U_i
  $h_j =$ the first k elements in U_j
  $k_\cap(i,j) = |h_i \cap h_j|$
  $h_{i,j}[k] =$ the first k elements in $h_i \cup h_j$
  $c = \frac{k_\cap(i,j)}{k} \frac{m(k-1)}{h_{i,j}[k]}$
else
  $c = 0$
  if $|U_i| > |U_j|$ **then** swap U_i and U_j
  $\beta = |U_i| - \tau$
  for $k = 0$ to β **do: if** $U_i[k] \in U_j$ **then** $c = c + 1$
  if $c = 0$ **then return** 0
  for $k = \beta$ to $|U_i| - 1$ **do: if** $U_i[k] \in U_j$ **then** $c = c + 1$
end if
return c

Performance Analysis. Algorithm 2 has a time complexity of $O(n + \mu^2 \min(l, \log(m)))$, where $\mu = |\{A[i] \,|\, |A[i]| \geq \tau\}|$.

5. SCALING SRP TO VERY LARGE SETTINGS

Recall that in SRP often n is a low dimensional vector with $n \ll m$. In this subsection we briefly describe how to extend DIRECT to handle cases where even n is very large (and still $n \ll m$). For example, let n be 10^6 and m be 10^{12}. A key aspect of DIRECT is that it leverages the sparse representation of the matrix (as against its complete dense representation) for speedup. However, when n is very large, even fitting the sparse representation of A into the memory may not be possible. To see why, even if there is only one non-zero value in *every column*, then we use $O(m)$ storage to even represent this matrix.

Interestingly, the similarity-joins based techniques proposed in § 4 do not require to completely materialize even sparse representation of A for estimating AA^T. Also, there are many scenarios where the user is interested in knowing the values of a subset of components of the reconstructed signals such as those corresponding to the largest values of the reconstructed signal. We now show how to adapt our algorithms to handle these scenarios.

Consider Algorithm 1 where the critical step is the first line. Algorithm 3 applies bottom-k sketch for the sets whose size is more than $\log m$. Thus, choosing the signature size in the bottom-k sketch to be in $O(\log m)$, Algorithm 3 needs at most $O(\log m)$ elements from *each row*. As a result, Line 1 of DIRECT needs a representation of size $O(n \log m)$ of A. For instance, in our example of $n = 10^6$ and $m = 10^{12}$, the size of the representative of A is only in the order of 1 million rows by 40 columns. Also, since AA^T is a sparse matrix, we only store the non-zero values of matrix t, rather than the complete n by n matrix. Line 2 is the multiplication of matrix A with X' whose dimensions are m by 1 followed by subtracting the n-dimensional result vector from the vector b. For this line, for each row of A, we use a sample of size $O(\log m)$ for the non-zero elements of the row, while using the values of X' as the sampling distribution. The result is a representation of size $O(n \log m)$ of A. Also, rather than loading the complete vector X' to the memory, in an iterative manner, we bring loadable buckets of it to the memory, update the calculation for that bucket, and move to the next one. In Line 4, t is the non-zero elements of AA^T and t' is a n by 1 vector, and finding the n by 1 vector ξ is doable, using methods like Gauss-Jordan. Finally, we only limit the calculations to the variables of interest, or even if the computation of all variables is required, in an iterative manner, we move a loadable bucket of them to the memory, compute their values, and move to the next bucket.

6. EXPERIMENTAL EVALUATION

6.1 Experimental Setup

Hardware and Platform. All our experiments were performed on a Macintosh machine with a 2.6 GHz CPU and 8GB memory. The algorithms were implemented using Python2.7 and Matlab.

Datasets. We conducted extensive experiments to demonstrate the efficacy of our algorithms over graphs with diverse values for number of nodes, edges and source-destination pairs. Recall that given a communication network, the size of the routing matrix A is parameterized by the number of edges and number of source-destination pairs - and not by the number of nodes and edges. The size of SRP that we tackle are 2-3 orders of magnitude larger than prior work such as [26]. Specifically, we used p2p dataset from SNAP repository of Stanford university[4]. The p2p dataset is a snapshot of the Gnutella network in August 2002 with 10876 nodes and 39994 edges. Nodes represent the hosts and the links represent the connection between the hosts. Each of the derived datasets is a subgraph of the overall p2p graph and was obtained by Forest Fire model [17]. The characteristics of each of these datasets dubbed *p2p-2* and *p2p-3* can be found in Table 1.

Constructing Traffic Matrices. Once we sample the network and obtain a connected graph, we consider all possible source destination pairs, i.e., #nodes×(#nodes−1), to be as individual flows.

[4]SNAP Dataset: https://snap.stanford.edu/data/p2p-Gnutella04.html

Table 1: Dataset Characteristics

Network	#Nodes	#Edges	#Source-Destination pairs
N_1	274	281	827
p2p-3	1438	7081	2M

For each source-destination pair we calculated the shortest path between them (network policies are not considered here as our algorithm is oblivious of the route chosen). Traffic matrix is a collection of all such routes in the following manner, each of the rows corresponds to an edge used in routing and each of the columns corresponds to a source-destination pair. Every cell, $c[i, j]$ is a '1' if edge$[i]$ is involved in routing traffic for source-destination$[j]$ else is assigned a value '0'. A visual glimpse of the routing matrix is given in Figure 2.

We used a *Pareto* traffic generation model, a popular stochastic model of the traffic flows for generating self-similar traffic observed in network communication [6]. The distribution is parametrized by a scale parameter x_m (set to 20) and a shape parameter α (set to 1). x_m is the minimum value of the distribution of traffic represented by the scale parameter while the shape parameter α indicates the 'steepness of the slope' of the distribution curve. The *prior* to the experiments (X') was obtained as a function of gravity model from [25].

6.2 Experimental Results

We compare the exact algorithm DIRECT with the baselines QP and WLSE [25]. The evaluation was conducted over small scale synthetic networks. Here we report the comparison results for N_1 (Table 1). Please refer to [1] for the complete experiment results. As shown in Figure 6, DIRECT significantly outperforms the baselines. In addition to comparing with these two baselines, for N_1, we also used compressive sensing [20] for estimating the values of the source-destination pairs. Since the objective in compressive sensing is the expensive l_0-optimization, even for our smallest setting N_1 it took 23.414 seconds.

We next evaluate the exact version of DIRECT and its approximate counterpart (using Algorithm 2) that leverages techniques from similarity joins to speed up the computation. We use DIRECT-E to refer to the exact version of DIRECT and DIRECT-A for its approximate version. Note that our algorithms take advantage of the sparse representation of matrix A and can perform the linear algebraic operations without materializing the entire matrix. We also evaluate the performance of our algorithms to two different threshold values of $(m/1000)$ and $(m/100)$, where m is the number of source-destination pairs. Choosing an appropriate threshold is often domain specific with larger thresholds providing better speedups. We compare the performance of the algorithms DIRECT-E and DIRECT-A through two metrics : performance and accuracy. We measure the former through execution time. We measure the accuracy of the signal reconstruction through *bucketized error* where we bucketize the source-destination pairs by the exact value of their flows and compute the error of the approximation algorithm within each bucket. The bucketization is often more illuminating for scenarios such as network traffic engineering where the signal exhibits a heavy tailed distribution and often the practitioner is interested in accurately estimating large flows. After finding the optimal flow assignments using the algorithm DIRECT-E, we sort the source-destination pairs in descending order, based on the amount of flow passing through them. For example, let a flow assignment by DIRECT-E be $\{(SD_1 : 3), (SD_2 : 24), (SD_3 : 7), (SD_4 : 75), (SD_5 : 5), (SD_6 : 12)\}$. The sorted SD pairs are $\{(SD_4 : 75), (SD_2 : 24), (SD_6 : 12), (SD_3 : 7), (SD_5 : 5), (SD_1 : 3)\}$. We then partition the SD pairs into 50 equal size buckets (each bucket contains 2% of SD pairs[5]). In the provided example, assume that we partition them into 3 buckets $B_1 : \{(SD_4 : 75), (SD_2 : 24)\}$, $B_2 : \{(SD_6 : 12), (SD_3 : 7)\}$, and $B_3 : \{(SD_5 : 5), (SD_1 : 3)\}$. For every SD pair, we consider the difference between the values computed by DIRECT-A and the one by DIRECT-E as the error of that SD pair, and compute the average for each bucket. In our example, let $\{(SD_1 : 5), (SD_2 : 24), (SD_3 : 6), (SD_4 : 79), (SD_5 : 5), (SD_6 : 11)\}$ be the assigned values by DIRECT-A. Then the average errors for the buckets B_1, B_2, and B_3 are 2, 1, and 1, respectively. It was observed in [25] that for many tasks in network traffic engineering such as routing optimization, even a relative error of *few* 10s of percent is considered tolerable.

p2p-3 (2M Source-Destination pairs) This network has 2M source-destination pairs with 7081 edges sampled from the *SNAP p2p* dataset. Figure 7 shows that DIRECT-E takes much as 1500 seconds to compute the exact solution. This is often prohibitive and simply unacceptable for many traffic engineering tasks. However, our approximate algorithms can provide the result in as little as 35 seconds. This is a significant reduction in execution time with a speedup of much as 97% of the running time of DIRECT-E. Figure 8 shows that the results are very close to the exact answer produced by DIRECT-E.

7. RELATED WORK

Linear Algebraic Techniques for Solving SRP: There has been extensive work on solving the system of linear equations using a wide variety of techniques such as computing the pseudoinverse of A [22] or performing Singular Value Decomposition (SVD) on A, and iterative algorithms for solving the linear system [22]. However, none of these methods scale for large-scale signal reconstruction problems. A key bottleneck in these approaches is often the computation of the pseudo inverse for matrix A. Note that any matrix B such that $ABA = A$ is defined as a pseudo inverse for A. It is possible to identify "the infinitely many possible generalized inverses" [22], each with its own advantages and disadvantages. Moore-Penrose Pseudo inverse (MPP) [21] is one of the most well-known and widely used pseudo inverse. MPP is the pseudo inverse that has the smallest Frobenius norm, minimizes the least-square fit in over-determined systems, and finds the shortest solution in the under-determined ones. However, none of the pseudo-inverse definitions suits our purpose of finding the solution X that minimizes the ℓ_2 distance from a prior. Furthermore, computing pseudo inverses is often done by SVD that is computationally very expensive.

8. CONCLUSION

In this paper, we investigated how a wide ranging problem of large scale signal reconstruction can benefit from techniques developed by the database community. Efficiently solving SRP has number of applications in diverse domains including network traffic engineering, astronomy, medical imaging etc. We propose an algorithm DIRECT based on the Lagrangian dual form of SRP. We identify a number of computational bottlenecks in DIRECT and evaluate the use of database techniques such as sampling and similarity joins for speeding them up without much loss in accuracy. Our experiments on networks that are orders of magnitude larger than prior work show the potential of our approach.

[5]We have found out the knee point of the cumulative flow is around 2%.

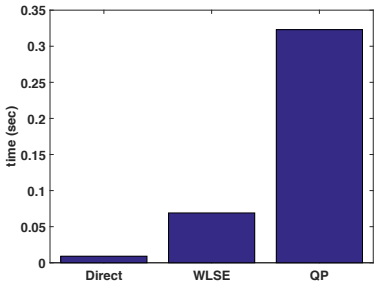

Figure 6: DIRECT **v.s. baselines in** $N_1 : n = 281$ **and** $m = 827$

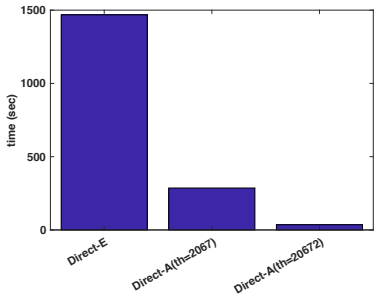

Figure 7: Execution time of DIRECT-E**,** DIRECT-A (τ**=2067), and** DIRECT-A (τ**=20672) in p2p-3**

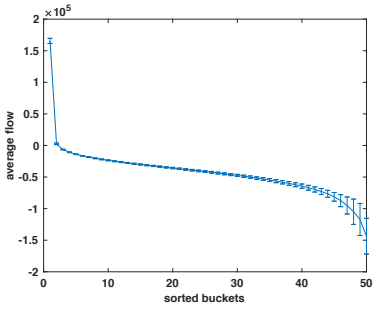

Figure 8: Absolute Error of the DIRECT-A ($\tau = 20672$) **in p2p-3**

9. ACKNOWLEDGMENTS

The work of Abolfazl Asudeh, Azade Nazi, Jees Augustine, and Gautam Das was supported in part by AT&T, the National Science Foundation under grant 1343976, and the Army Research Office under grant W911NF-15-1-0020. Nan Zhang was supported in part by the National Science Foundation, including under grants 1343976, 1443858, 1624074, 1760059, and by the Army Research Office under grant W911NF-15-1-0020.

10. REFERENCES

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

[2] Y. Awatsuji, Y. Wang, P. Xia, and O. Matoba. 3d image reconstruction of transparent gas flow by parallel phase-shifting digital holography. In *WIO*, 2016.

[3] K. Beyer, R. Gemulla, P. J. Haas, B. Reinwald, and Y. Sismanis. Distinct-value synopses for multiset operations. *Communications of the ACM*, 52(10):87–95, 2009.

[4] A. Z. Broder. On the resemblance and containment of documents. In *SEQUENCES*, pages 21–29. IEEE, 1997.

[5] J. Cao, D. Davis, S. Vander Wiel, and B. Yu. Time-varying network tomography: router link data. *Journal of the American statistical association*, 95(452):1063–1075, 2000.

[6] B. Chandrasekaran. Survey of network traffic models. *Waschington University in St. Louis CSE*, 567, 2009.

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

[8] E. Cohen and H. Kaplan. Tighter estimation using bottom k sketches. *PVLDB*, 1(1):213–224, 2008.

[9] V. F. Farias, S. Jagabathula, and D. Shah. A nonparametric approach to modeling choice with limited data. *Management science*, 59(2):305–322, 2013.

[10] Y. Gong. Identifying p2p users using traffic analysis. 2005. `www.symantec.com/connect/articles/identifying-p2p-users-using-traffic-analysis`.

[11] P. Grangeat and J.-L. Amans. *Three-dimensional image reconstruction in radiology and nuclear medicine*, volume 4. Springer Science & Business Media, 2013.

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

[13] P. C. Hansen. *Rank-deficient and discrete ill-posed problems: numerical aspects of linear inversion*. SIAM, 1998.

[14] W. T. Hrinivich, D. A. Hoover, K. Surry, C. Edirisinghe, D. D'Souza, A. Fenster, and E. Wong. Ultrasound guided high-dose-rate prostate brachytherapy: Live needle segmentation and 3d image reconstruction using the sagittal transducer. *Brachytherapy*, 15:S195, 2016.

[15] S. V. Kalinin, E. Strelcov, A. Belianinov, S. Somnath, R. K. Vasudevan, E. J. Lingerfelt, R. K. Archibald, C. Chen, R. Proksch, N. Laanait, et al. Big, deep, and smart data in scanning probe microscopy, 2016.

[16] P. Kuchment and F. Terzioglu. 3d image reconstruction from compton camera data. *arXiv:1604.03805*, 2016.

[17] J. Leskovec, J. Kleinberg, and C. Faloutsos. Graphs over time: densification laws, shrinking diameters and possible explanations. In *SIGKDD*, pages 177–187. ACM, 2005.

[18] Z. Liu, Z. Shi, M. Jiang, J. Zhang, L. Chen, T. Zhang, and G. Liu. Using MC algorithm to implement 3d image reconstruction for yunnan weather radar data. *Journal of Computer and Communications*, 5(05), 2017.

[19] R. Massey, J. Rhodes, R. Ellis, N. Scoville, A. Leauthaud, A. Finoguenov, P. Capak, D. Bacon, H. Aussel, J.-P. Kneib, et al. Dark matter maps reveal cosmic scaffolding. *arXiv preprint astro-ph/0701594*, 2007.

[20] D. Needell and J. A. Tropp. Cosamp: Iterative signal recovery from incomplete and inaccurate samples. *Applied and Computational Harmonic Analysis*, 26(3), 2009.

[21] R. Penrose. A generalized inverse for matrices. In *Mathematical proceedings of the Cambridge philosophical society*, volume 51, pages 406–413, 1955.

[22] C. R. Vogel. *Computational methods for inverse problems*. SIAM, 2002.

[23] Y. Wiaux, L. Jacques, G. Puy, A. M. Scaife, and P. Vandergheynst. Compressed sensing imaging techniques for radio interferometry. *Monthly Notices of the Royal Astronomical Society*, 395(3):1733–1742, 2009.

[24] G. L. Zeng. 3d image reconstruction. In *Medical Image Reconstruction*, pages 87–123. Springer, 2010.

[25] 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, 2003.

[26] Y. Zhang, M. Roughan, C. Lund, and D. Donoho. An information-theoretic approach to traffic matrix estimation. In *SIGCOMM*, pages 301–312. ACM, 2003.

[27] Y. Zhu, Z. Li, H. Zhu, M. Li, and Q. Zhang. A compressive sensing approach to urban traffic estimation with probe vehicles. *IEEE Transactions on Mobile Computing*, 12(11):2289–2302, 2012.