A Sketch-Based Distance Oracle for Web-Scale Graphs

Atish Das Sarma
Georgia Institute of Technology
Atlanta, GA, USA
atish@cc.gatech.edu

Sreenivas Gollapudi
Microsoft Research
Mountain View, CA, USA
sreenig@microsoft.com

Marc Najork
Microsoft Research
Mountain View, CA, USA
najork@microsoft.com

Rina Panigrahy
Microsoft Research
Mountain View, CA, USA
rina@microsoft.com

ABSTRACT

We study the fundamental problem of computing distances between nodes in large graphs such as the web graph and social networks. Our objective is to be able to answer distance queries between pairs of nodes in real time. Since the standard shortest path algorithms are expensive, our approach moves the time-consuming shortest-path computation offline, and at query time only looks up precomputed values and performs simple and fast computations on these precomputed values. More specifically, during the offline phase we compute and store a small "sketch" for each node in the graph, and at query-time we look up the sketches of the source and destination nodes and perform a simple computation using these two sketches to estimate the distance.

Categories and Subject Descriptors

G.2.2 [**Graph Theory**]: Graph algorithms, path and circuit problems

General Terms

Algorithm, Performance

Keywords

Algorithms, Sketching, Embedding, Distance Computation, Shortest Path

1. INTRODUCTION

Large graphs have become a common tool for representing real world data. We now routinely interact with search engines and social networking sites that make use of large web graphs and social networks behind the scenes. Many of these interactions happen in real time where users make some kind of a request or a query that needs to be serviced almost instantaneously. Since user interactions should have low latency, one method to handle expensive operations is to do them offline as a precomputation and store the data so that they can be obtained quickly when required for a real-time operation. For example, PageRank consists of an expensive eigenvector computation over the web graph that can be performed offline and a simple online lookup of the PageRank score for each result.

One fundamental operation on large graphs is finding shortest paths between pairs of nodes. This problem is not only a common building block in many algorithms, but is also a meaningful operation in its own right. For example, in a social network one may be interested in finding the shortest sequence of friends that connects one to a celebrity. In the web graph, a short sequence of links between two URLs may indicate a certain degree of relatedness between the two pages [13, 20]. However, given the large size of these graphs, shortest-path computation is challenging. Running Dijkstra's well known shortest-path algorithm [8] on a web graph containing tens of billions of nodes and trillions of edges would take several hours, if not days. Moreover, it is not feasible to store a web-scale graph in the main memory of a single machine. Even in a distributed setting, if the computation is parallelized, the sequentially dependent nature of Dijkstra's computation would require huge amounts of communication. Furthermore, in a real-time computation, only a small amount of resources – memory accesses and CPU cycles – are available for a single shortest distance query. As we would like to do this in real time with minimal latency, it becomes important to use small amounts of resources per distance query. In this paper, we study the problem of estimating distances between two nodes in a large graph in real time.

One approach is to perform a one-time offline computation. A straightforward brute force solution would be to compute the shortest paths between all pairs of nodes offline and to store the distances on disk. In this setting, answering a shortest-path query online requires a single disk lookup; however, the space requirement is quadratic in the number of nodes in the graph. For a web graph containing billions of nodes, this would be simply infeasible. To reduce the space complexity, our algorithms store some auxiliary information with each node that can facilitate a quick distance computation online in real time. This auxiliary information is then used in the online computation that is performed for every request or query. One can view this auxiliary information as a *sketch* of the neighborhood structure of a node that

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
WSDM'10, February 4–6, 2010, New York City, New York, USA.
Copyright 2010 ACM 978-1-60558-889-6/10/02 ...$10.00.

is stored with each node. Simply retrieving the sketches of the two nodes should be sufficient to estimate the distance between them. Two properties are crucial for this purpose: First, these sketches should be reasonably small in size so that they can be stored with each node and accessed for any node at run time. Second, there needs to be a simple algorithm that, given the sketches of two nodes, one can estimate the distance between them quickly. As the computation of the sketches is an offline computation, one can afford to spend more resources on this one-time preprocessing.

Computing sketches offline for a large collection of objects to facilitate online computations has been studied extensively and is also used in many applications. For example, several search engines compute a small sketch of documents to detect near-duplicate documents at query time; see [5, 10, 4] and references therein. This eliminates the need to compare large documents, which is time consuming. Instead it is achieved by comparing short sketches of these documents and measuring the similarity between these sketches.

Computing sketches for the specific purpose of distance computation is also called distance labeling. Some papers that study the problem of the size of labels required with each node to allow distance computation include Gavoille et al. [14], Katz et al. [16], and Cohen et al. [7].

The field of metric embedding deals with mapping a set of points from a high-dimensional space to a low-dimensional space, such that the distortion is minimized. If each point can be projected onto a small number of dimensions such that distances are approximately preserved, one can store the small dimensional vector as a sketch. The classic result of Bourgain [3] shows how such an embedding can be achieved for certain distance metrics.

Another line of work in estimating distances is the study of *spanner* construction. A spanner is a sparse subgraph of the given graph, such that the distance on this sparse graph approximates the actual distance, for any pair of points. Although spanners take small space, they do not exactly provide a sketch for each node; thus the online algorithm for estimating distance may take a long time. Some theoretically efficient algorithms for spanners are presented by Feigenbaum et al. [12], and Baswana [2]. Other fundamental results in this area include Bartal [1] and Fakcharoenphol et al. [11].

Cohen et al. [6] proposed an approximate distance scheme using 2-hop covers of all paths in a directed (or undirected) graphs. However, finding the near optimal 2-hop cover of a given set of paths is expensive in a large graph. Moreover, the size of such a cover can be as large as $\Omega(n\sqrt{m})$, making their scheme quite hard to implement on large graphs.

Several studies, for example the ones by Goldberg et al. [15, 9], have focused on answering exact shortest path queries on road networks. These algorithms make use of a small set of precomputed landmarks and shortcuts and use them at query time to connect a source/destination pair. Landmarks and shortcuts are chosen very carefully, using algorithms that are specialized to the structure of road networks. Our algorithms can be viewed as using a randomly sampled set of landmarks; as such, it can be taken to mean that even a simple algorithm that makes use of randomly sampled "landmarks" works very well on other complex graphs such as the web graph.

2. OUR CONTRIBUTION

In this work, we engineer algorithms for computing such sketches and demonstrate how they can be used to estimate distances between arbitrary pairs of nodes, in real time. While there is not much empirical work on computing sketches for distances, there are the aforementioned theoretical studies using embeddings [3, 1, 11] and spanners [12, 19, 2] in the algorithms literature. All of these algorithms provably work only on undirected graphs, and some are complicated and probably impractical. The classical result by Bourgain [3] shows how one can project a graph onto a low dimensional space, giving a small sketch that approximates distances to a factor of $O(\log n)$. Matousek [17] later showed that the same algorithm can be used to get a $2c-1$ factor approximation using sketches of size $\tilde{O}(n^{1/c})$. However, we find in our experiments that Bourgain's algorithm performs very poorly in estimating the distance. On the other hand, we propose an algorithm that is essentially a simplification of the algorithm by Thorup and Zwick [19] and provides the same theoretical guarantee: our algorithm approximates distances within a factor $2c-1$ by using sketches of size $\tilde{O}(n^{1/c})$ (note that for $c = \log n$ this gives a $O(\log n)$-factor approximation to the distances using sketches of size $O(\log n)$). Furthermore, our algorithm is simpler to implement. While both algorithms sample seed sets of different sizes and find the closest seed in each seed set, Thorup's and Zwick's algorithm needs to store additional data in the sketch of a node, namely all IDs of nodes in a seed set that are closer than the nearest seed in the next seed set when ordered by decreasing size. This adds more complexity to the offline precomputation and also introduces some additional checks beyond what we do in the online step. In this paper, we demonstrate that the additional data in the sketch is not required, and simply keeping the seed ID along with the distance is sufficient.

We further improve on existing work by extending our algorithms to directed graphs and evaluating them on large graphs. We find in our experiments that even on a large web graph, our algorithm gives very good estimates for both directed and undirected distances – the distances are usually accurate to within an additive error of 2 to 3. The fact that our algorithm performs well for directed distances is surprising as there is no known sketching algorithm for directed distances; in fact, it is known that this is impossible in the worst case for arbitrary directed graphs, which suggests that the web graph has some special properties. Understanding the gap between the theoretical guarantee and the experimental observation is an interesting area for future investigation.

The essential idea behind our algorithm is the following: In the offline computation, sample a small number of sets of nodes in the graph (sets of seed nodes). Then, for each node in the graph, find the closest seed in each of these seed sets. The sketch for a node simply consists of the closest seeds, and the distance to these closest seeds. Then, in the online computation, one can use the distance to this closest seed to estimate the distance between a given pair of nodes. One method to do this is to check if there is a common node between the two sketches. Given a pair of nodes u and v one can estimate the distance between them by looking for a common seed in their sketches. If w is a common seed in the sketch of u and v then the distance can be estimated by adding up the distances to the common seed w. We also

note that our algorithm produces an upper bound on the actual distance; whereas Bourgain's algorithm produces a lower bound.

Experimentally we observed that our algorithm performs very well for estimating both directed and undirected graphs. We conduct experiments on a 2002 crawl of the web graph. We compare our estimates with precise distances computed by Dijkstra's algorithm between sampled pairs of nodes. The actual undirected distances are in the range of 1 and 15, and the directed distances (whenever connected by a directed path) are in the range of 1 and 100. We sample pairs from each of these distances for examining the performance. For all pairs u, v with actual distance $d(u, v)$, we compute the median of the distance estimate obtained by each of our algorithms. We do this independently for undirected and directed distances.

3. OUR ALGORITHMS

We will now describe the algorithm to compute sketches for each node that can be used to perform online distance computations between a pair of nodes u and v. For simplicity, we only work with undirected graphs in this section, and generalize to directed graphs in Section 4.

We denote a graph by G, its nodes by V, edges by E, and number of nodes $|V|$ by n. The distance between u and v is denoted by $d(u, v)$. Our goal is to preprocess and store this graph in such a way that given any pair of nodes u and v, we are able to estimate the distance $d(u, v)$ in real time using a small amount of computation. It is important to keep in mind that a web-scale graph contains tens of billions of nodes and trillions of edges. The precomputation consists of computing a sketch Sketch$[u]$ for each node u. The real-time computation of the distance between u and v should involve only reading Sketch$[u]$ and Sketch$[v]$.

We will use $\tilde{d}(u, v)$ to denote an estimate for $d(u, v)$ obtained by a sketching algorithm. Another notation that we use in the paper is $d(u, S) = \min_{w \in S} d(u, w)$, which is the shortest distance between u and a set of nodes $S \subseteq V$.

Algorithm Offline-Sketch

As stated earlier, the essential idea in our algorithm is to sample a set of seed nodes, and store the closest seed from every node along with its distance. This can be done efficiently by just one Breadth First Search (BFS) from the seed set (note that it is even possible to find the nearest node in the set from each node by passing node IDs appropriately while performing the BFS). Given a node u and a seed set S, the $s \in S$ nearest to u is referred to as the seed of u in S. We do this for $\log n$ sets of sizes that are different powers of 2 in function Offline-Sample (see Algorithm 1) which returns the nearest seed and the distance to it in each of these sets. Function Offline-Sketch (see Algorithm 2) repeats this k times – each time using different random sets – and returns the union of the samples. All of this is done offline and stored as Sketch$[u]$ for each node u in the graph.

In order to help in estimating the distance between a source vertex u and destination vertex v, the seed vertex must lie on a path between u and v. In order to lie on a directed path, a vertex must have non-zero in- and out-degree. Thus, we should consider only such vertices as candidates for a seed set. In the undirected case, we should consider only vertices with degree of at least 2.

Algorithm 1 Offline-Sample(G)

Input: An undirected graph G and a node u in the graph.
Output: Sample$[u]$, A set of nearest seeds and their distances for each node u.

1: Let V denote the set of vertices, let $r = \lfloor \log |V| \rfloor$. Sample $r + 1$ sets of sizes $1, 2, 2^2, 2^3, ..., 2^r$ respectively. In each set, the samples are chosen uniformly at random from all nodes. (As stated before, when sampling nodes from V we may ignore nodes with degree < 2 as they cannot be in the shortest path between a source and a destination). Call the sampled sets $S_0, S_1, \ldots, S_r$.
2: For each node u, for all these sets S_i, compute (w_i, δ_i) where w_i is the closest node to u in S_i and $\delta_i = d(u, w_i) = d(u, S_i)$.
This can be done for all nodes u efficiently with just one BFS from each set S_i. Sketch$[u]$ = $\{(w_0, \delta_0), \ldots, (w_r, \delta_r)\}$

Algorithm 2 Offline-Sketch(G)

Input: An undirected graph G and a node u in the graph.
Output: Sketch$[u]$, the sketch of node u.

1: Run Offline-Sample(G) k times, each time sampling independently at random.
2: Sketch$[u]$ = Union of the Sample$[u]$ returned by each of the k runs.

The above candidate selection rule applies to any graph. There are other rules that may improve the accuracy of our algorithms further (for example, biasing the sampling process towards high-degree nodes), but they are dependent on the topology of the graph (for example, biasing toward high-degree nodes is a bad strategy in a "dumbbell" graph); therefore, we did not consider them in the paper.

Algorithm Online-Common-Seed

We now present our main algorithm that estimates distances using the sketches. Given nodes u and v, algorithm Online-Common-Seed (see Algorithm 3) approximates the distance between them by looking at the distance from u and v to any node w occurring in both Sketch$[u]$ and Sketch$[v]$. The length of the shortest path from u to v through w is $d(u, w) + d(w, v)$; note that both $d(u, w)$ and $d(w, v)$ are contained in the sketches and do not need to be computed. We take the minimum of $d(u, w) + d(w, v)$ over all such common seeds w to be the estimated distance $\tilde{d}(u, v)$. It is easily shown that this distance estimated by Online-Common-Seed is always an upper bound on the actual distance.

Observation 3.1. *The estimated distance $\tilde{d}(u, v)$ from* Online-Common-Seed(u,v) *is an upper bound of the actual distance $d(u, v)$.*

Proof. The algorithm considers various nodes w that are in the intersection of the two sketches of u and v. The sketches contain exact distances $d(u, w)$ and $d(w, v)$ between each w and u and v, respectively. By triangle inequality, $d(u, w) + d(w, v) \geq d(u, v)$. Notice that we take the minimum sum over several w, but for each of these nodes, the triangle inequality holds. The observation follows. □

Algorithm 3 ONLINE-COMMON-SEED(u,v)

Input: Nodes u and v from G between which the distance is to be estimated.
Output: An estimate of $d(u, v)$.

1: Obtain SKETCH[u] and SKETCH[v].
2: Find the set of common nodes w in SKETCH[u] and SKETCH[v]. Note that there is at least one w for undirected G assuming that G is connected, as we perform a BFS from at least one set of size 1.
3: For each common node w, compute $d(u, w)$ and $d(w, v)$.
4: Return the minimum of $d(u, w) + d(w, v)$, taken over all such common w's. If no common seed w is present, then output ∞.

THEOREM 3.2. *The estimated distance $\tilde{d}(u, v)$ returned by* ONLINE-COMMON-SEED(u,v) *for $k = \tilde{\Theta}(n^{1/c})$ in the sketch computation gives with high probability*[1] *a $2c-1$ approximation to the actual distance for all pairs; $d(u, v) \leq \tilde{d}(u, v) \leq (2c - 1)d(u, v)$.*

PROOF. Let $d = d(u, v)$. Let A_r, B_r denote balls of radius rd around u and v respectively; that is, all nodes within distance at most rd around u and v.

Consider the points in $A_r \cup B_r$ and $A_r \cap B_r$. If one of the seed sets S is such that it has exactly one seed w in the union $A_r \cup B_r$ which is also in the intersection $A_r \cap B_r$, then w is a common seed in the sketch of u and v. This is because it is the closest seed to both u and v. We will argue that such a seed set exists and results in a common seed in the sketch that is at distance at most $d \log n$ from both u and v. For simplicity, let us consider the case when $c = \log n$. Precisely, we observe that if $\frac{|A_r \cap B_r|}{|A_r \cup B_r|}$ is at least some constant (say 1/2), then when we take seeds with probability $\frac{1}{|A_r \cup B_r|}$, there is a constant chance that exactly one seed is present in the union which also happens to be in the intersection. If seeds are sampled with probability $\frac{1}{|A_r \cup B_r|}$, this event happens with probability at least $1/(2e)$ since with probability $1/e$ there is exactly one seed and further with probability 1/2 it lies in the intersection. Since we are trying seed set sizes that are different powers of 2, the probability will be constant for the closest power of 2. Thus, if $\frac{|A_r \cap B_r|}{|A_r \cup B_r|} > 1/2$ for any i in the range $1..\log n$, then there is a constant probability of finding a common seed within distance $d \log n$ from both u and v. If not, this means $\frac{|A_r \cap B_r|}{|A_r \cup B_r|} \leq 1/2$ for all $1 \leq i \leq \log n$.

But note that $A_r \cup B_r \subseteq A_{r+1} \cap B_{r+1}$ since the set on the left hand side contains points at distance at most rd from u or v, and further since $d(u, v) = d$, these points are at distance at most $(r + 1)d$ from both u and v implying that they are all present in $A_{r+1} \cap B_{r+1}$. So if $\frac{|A_r \cap B_r|}{|A_r \cup B_r|} \leq 1/2$ for all $1 \leq i \leq \log n$, this means $|A_{r+1} \cup B_{r+1}| > 2|A_r \cup B_r|$, implying $|A_{\log n} \cup B_{\log n}| > n$ which is impossible. Thus there must be a value of r such that $\frac{|A_r \cap B_r|}{|A_r \cup B_r|} \leq 1/2$. Since we try each size k times, for constant k, we can make the probability of failure negligible. The same proof generalizes to arbitrary c; we show that is some i, $1 \leq r \leq c$, such that $\frac{|A_r \cap B_r|}{|A_r \cup B_r|} \geq n^{-1/c}$; if we repeat $k = \tilde{\Theta}(n^{1/c})$ times, we succeed with high probability, giving a $2c$ approximation.

[1] with high probability means with probability $1 - 1/n^{\Omega(1)}$

Algorithm 4 ONLINE-BOURGAIN(u,v)

Input: Nodes u and v from G between which the distance is to be estimated.
Output: An estimate of $d(u, v)$.

1: Obtain SKETCH[u] and SKETCH[v].
2: For each seed set S, extract $d(u, S)$ from SKETCH[u] and $d(v, S)$ from SKETCH[v], and compute $|d(u, S) - d(v, S)|$.
3: Return the maximum $|d(u, S) - d(v, S)|$ over all seed sets S.

This can be strengthened to give a $2c - 1$ approximation by looking at the sets $A_r \cup B_{i-1}$ and $A_r \cap B_{r-1}$ – note that $|A_r \cap B_{r-1}| = 1$ and for any point in the intersection the sum of the distances from u and v is at most $i \cdot d + (r - 1)d = (2r - 1)d \leq (2c - 1)d$. □

Algorithm Online-Bourgain

We compare the performance of our algorithm to that of Bourgain's well-known technique that embeds some metric space into a low dimensional space. We describe this in ONLINE-BOURGAIN (see Algorithm 4). We note that the same sketches are used as before. However, instead of finding nodes in the intersection of both sketches, we find the distance from v to all the seed sets and similarly the distance from u to all the seed sets. The L_∞-norm of the difference of these two vectors gives a lower bound on the actual distance between u and v.

Furthermore, Bourgain also proves that if we set k to $\Theta(\log n)$ (where k is the number of sets picked for each of the $O(\log n)$ sizes), then the distances are accurate up to constant factor of $\log n$. By using seed sets of size powers of 2 and using k sets of each size, one can prove theoretical guarantees on the quality of estimates. We present a slight alteration that can be extended to directed graphs. While the intuition carries over to directed graphs, unfortunately one can no longer prove a $O(\log n)$ approximation ratio. However, we present the simple observation that the returned distance estimate $\tilde{d}(u, v)$ is a lower bound on the actual distance $d(u, v)$. Notice that in the algorithm, we consider the absolute value of $d(u, S) - d(v, S)$ for undirected graphs. This is justified by showing in our proof that both directions impose a lower bound, and hence the absolute value does too.

OBSERVATION 3.3. *The distance estimate $\tilde{d}(u, v)$ returned by* ONLINE-BOURGAIN(u,v) *is a lower bound on the actual distance $d(u, v)$ from u to v.*

PROOF. The proof follows essentially from triangle inequality. We will show that for any set S of nodes $|d(u, S) - d(v, S)| \leq d(u, v)$. It is sufficient to show that $d(u, S) - d(v, S) \leq d(u, v)$ (as by symmetry this will also imply that $d(v, S) - d(u, S) \leq d(u, v)$). To see this, let $d(v, S) = d(v, v')$ where v' is the closest node to v in S. $d(u, v') \leq d(u, v) + d(v, v')$ by triangle inequality, which gives $d(u, v') \leq d(u, v) + d(v, S)$. So $d(u, v) \geq d(u, v') - d(v, S) \geq d(u, S) - d(v, S)$ since u' is the closest node to u in S. Now the estimate $\tilde{d}(u, v)$ is obtained by taking the maximum of $|d(u, S) - d(v, S)|$ over different sets S used in the offline phase. But since for each S, the difference is bounded by $d(u, v)$, the maximum is also bounded by $d(u, v)$ □

Matousek [17] proved that Bourgain's algorithm ONLINE-BOURGAIN gives a $2c-1$-approximation to distances using sketches of size $\tilde{O}(n^{1/c})$ However, note that this result holds only for undirected graphs.

THEOREM 3.4. *The distance estimate $\tilde{d}(u,v)$ computed by* ONLINE-BOURGAIN(u,v) *for $k = \Theta(n^{1/c})$ in the sketch computation is with high probability an $O(2c-1)$ factor approximation; $\Omega(\frac{d(u,v)}{2c-1}) \leq \tilde{d}(u,v) \leq d(u,v)$.*

PROOF. Given in [3] □

4. GENERALIZATION TO DIRECTED GRAPHS

The algorithms presented in the previous section can be easily extended to the directed case. However, there are no known theoretical guarantees except for the upper and lower bounds we have shown. We state the main differences in the directed algorithms as compared to the undirected algorithms by rewriting the changed subroutines.

Modification to Online-Common-Seed

As before, the accuracy of the upper bound improves with the number of sets sampled in the offline sketch computing phase. It is important to choose sets of different sizes (to capture the right distance) as well as multiple sets of each size (to reduce the sensitivity of finding exactly the same nearest node from both u and v).

In the directed version of OFFLINE-SAMPLE, we compute two sketches for each vertex u, one capturing distances *to* the closest seeds, and one capturing distances *from* the closest seeds. For each sampled S, we find w_1 in S so that $d(u,w_1) = d(u,S)$ (i.e. $d(u,w_1)$ is minimized over $w_1 \in S$), and w_2 such that $d(w_2,u) = d(S,u)$ (i.e. $d(w_2,u)$ is minimized over $w_2 \in S$). One can obtain this efficiently for all nodes u with just two BFS runs from S. One run starts with S and iterates by traversing along incoming edges. The other run uses the outgoing edges.

The online distance computation algorithm then considers all w in the sketches of u and v. However, to compute the directed distance $d(u,v)$, we consider all w in the sketch corresponding to the distance from u to S and for v, the sketch corresponding to the distance from S to v. The distance estimate is then computed as before, $\tilde{d}(u,v) = d(u,S) + d(S,v)$.

The proof of the upper bound for the directed case follows in the same way as the undirected case, using triangle inequality for directed distances.

Modification to Online-Bourgain

The algorithm for finding the lower bound estimate of $d(u,v)$ in directed graphs is a minor modification of the algorithm for undirected graphs. We describe it in ONLINE-BOURGAIN-DIRECTED(u,v) (see Algorithm 5).

One difference is that when we consider $d(u,S) - d(v,S)$, we use the maximum between this and 0 and not the absolute value. This is crucial since directed graph distances do not satisfy the symmetry property of a metric. We did not have to consider this in the undirected algorithm as there always was a path between every node and every S and distances are symmetric. However, in directed graph, there may not be a path from/to a node to/from the set (as the input graph need not be strongly connected). Therefore, the quantity we are subtracting may turn out to be ∞ (so taking max eliminates obtaining negative values). Notice that we do not need to worry about the distance becoming positive ∞ (this can be verified from the correctness of the lower bound observation). Another minor difference is that we compute two quantities here, $d(S,v) - d(S,u)$ as well as $d(u,S) - d(v,S)$. This only gives us a stronger lower bound, as we shall show that both quantities independently are lower bounds.

Algorithm 5 ONLINE-BOURGAIN-DIRECTED(u,v)

Input: Nodes u and v from G between which the distance is to be estimated.
Output: An estimate of $d(u,v)$.

1: Obtain SKETCH$[u]$ and SKETCH$[v]$.
2: For each set distance in sketch corresponding to a set S, extract $d(u,S)$, $d(S,u)$ and $d(v,S)$, $d(S,v)$.
3: Compute $\max\{0, d(S,v) - d(S,u), d(u,S) - d(v,S)\}$ and find the maximum of this quantity over distances corresponding to all sets used to compute sketch from outgoing edge and incoming edge BFSs.
4: Maximum of the above two steps is returned as $\tilde{d}(u,v)$.

OBSERVATION 4.1. *The distance estimate $\tilde{d}(u,v)$ returned by* ONLINE-BOURGAIN-DIRECTED(u,v) *is a lower bound on the actual distance $d(u,v)$ from u to v.*

PROOF. We need to show that the triangle inequality holds for both steps. Since we are taking the max with 0, it is sufficient to show that both $d(S,v) - d(S,u) \leq d(u,v)$ and $d(u,S) - d(v,S) \leq d(u,v)$. It is important to maintain the directed distances here. Rearranging, these follow from triangle inequality, using the sequence of arguments described in Observation 3.3, for every set S. Since we are taking the maximum over certain sets to compute $\tilde{d}(u,v)$, this is a lower bound of $d(u,v)$. □

In the following section, we compare the performance of the algorithms on large web graphs.

5. EXPERIMENTS

Our experiments are performed on a large crawl of the web graph. We present some basic statistics here. The crawl was conducted in 2002 and is the result of a breadth-first search crawl starting at `www.yahoo.com`. We conducted our experiments on prefixes of the crawl. The number of crawled web pages is 65,581,675 and the number of distinct URLs is 419,545,168. This means that there are about five times as many nodes in the uncrawled "frontier" as in the explored part of the graph. The total number of edges in the graph is 2,371,215,893. The average out-degree of crawled pages is 36.16, and the average in-degree of all pages (whether crawled or not) is 5.65. The maximum out-degree and in-degree are 27,764 and 1,402,576 respectively.

In order to evaluate our algorithm on a given graph, we sampled 100 nodes at random. For each node v, we computed the distance between v node and all other nodes u. In the directed case, we computed the distance from all nodes u to v (which is ∞ if there is no path from u to v). We grouped the u's by distance and selected (up to) 10 nodes at random from each group, i.e. for each given distance. This left us with a test set of (u,v,d) triples for each given graph.

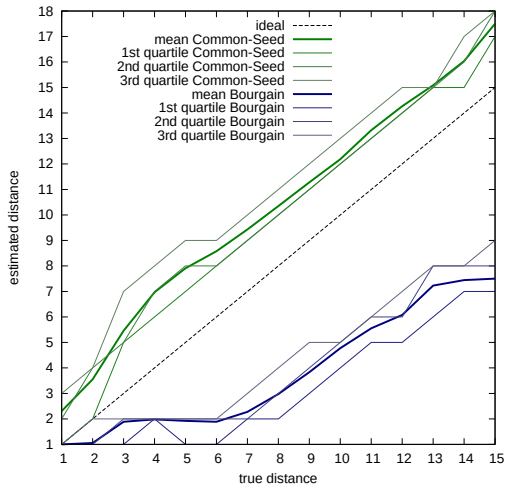

Figure 1: Estimates of undirected distances with $k = 1$

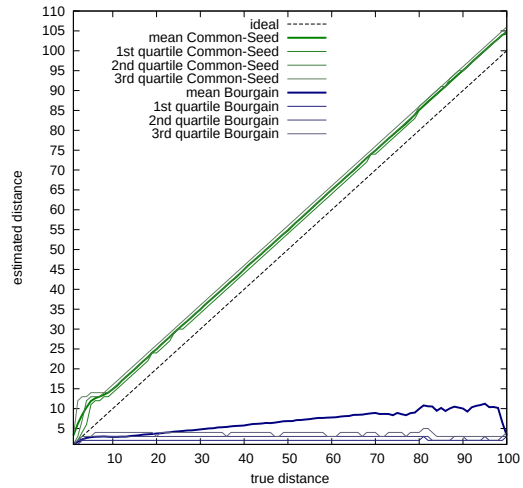

Figure 2: Estimates of directed distances using with $k = 1$

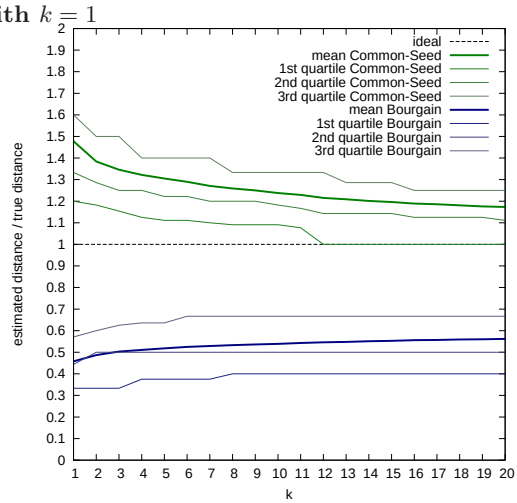

Figure 3: Estimates of undirected distances using as a function of k

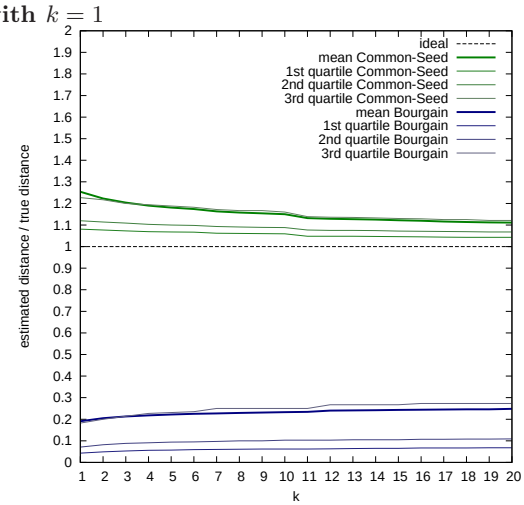

Figure 4: Estimates of directed distances using as a function of k

In order to conduct our experiments, we used the Scalable Hyperlink Store [18], a distributed system that partitions the web graph across many SHS servers, with each server maintaining a portion of the graph in main memory. Our implementation consists of an offline phase and an online phase. During the offline phase, we select the $\log n$ seed sets of exponentially increasing sizes and compute distances between every node in the graph and its closest seed using Dijkstra's algorithm. For each seed set, we output a temporary file consisting of a `seedid, distance` pair for each vertex in the graph. At the conclusion of the offline phase, we merge all the temporary files into a single sketch file consisting of $\log n$ pairs for each vertex. In directed graphs, we run the offline phase twice – once to compute distances from seeds to vertices, and once from vertices to seeds. To implement Online-Common-Seed, we repeat the offline phase k times. Finally, we merge the k (or $2k$) files into a single file.

During the online phase, we read the `seedid, distance` vectors of both the source and the destination vertex from disk. So, each query involves two disk seeks. Given that a disk seek takes several milliseconds while the subsequent processing of the sketches takes only microseconds, our algorithm is as fast in practice as Thorup's and Zwick's algorithm, despite the fact that from a theoretical complexity perspective, their algorithm has better time bounds for the online computation, requiring $O(c)$ time for a $(2c-1)$-approximation while ours requires $\tilde{O}(n^{1/c})$.

Single seed set sampling (k=1)

In our data set, true pairwise undirected distances vary between 1 and 15. We sample pairs to cover most of these distances and then query their sketches to find approximate distances. The directed graph is not strongly connected. But for pairs where there is a directed path, the distance spans the range of 1 to 100.

Figure 1 plots the estimates obtained from the algorithms Online-Common-Seed and Online-Bourgain respectively. In both plots, the x-axis denotes the actual distance of the sampled pair, the y-axis denotes the estimated distance. Therefore, the $x = y$ line corresponds to a perfect prediction of the actual distance. We show curves for the mean, median, 75th percentile and 25th percentile.

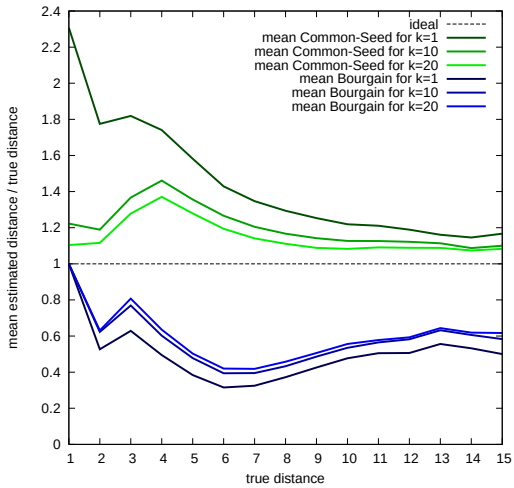

Figure 5: Ratio of estimated distance to true distance for different values of undirected true distance

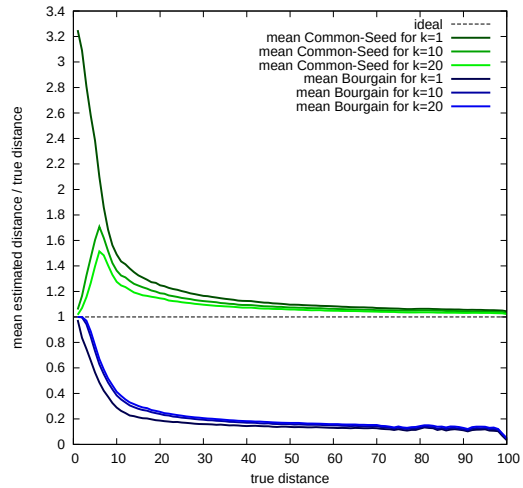

Figure 6: Ratio of estimated distance to true distance for different values of directed true distance

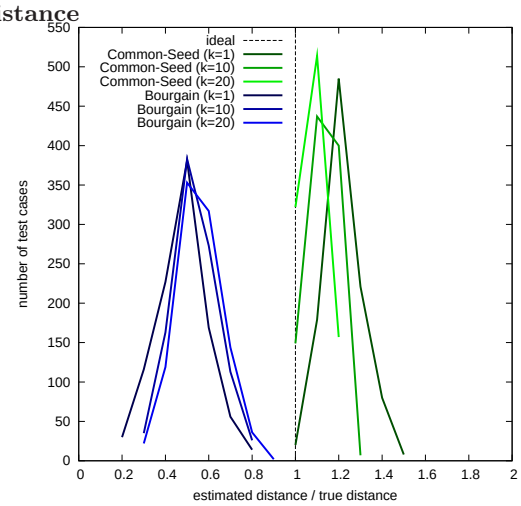

Figure 7: Distributions of the ratio of estimated distance to true distance for undirected distance $d = 10$

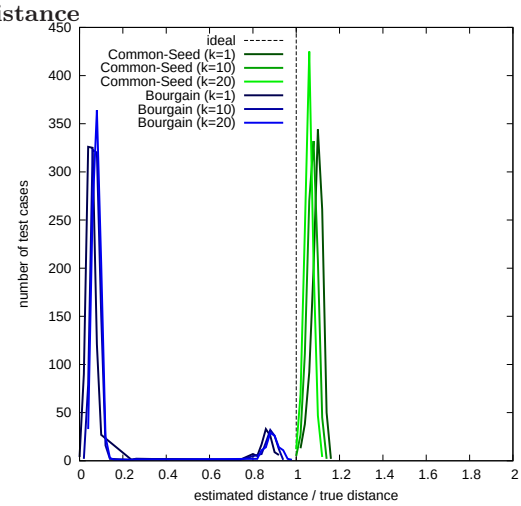

Figure 8: Distributions of the ratio of estimated distance to true distance for directed distance $d = 50$

The same plot for directed distances is shown in Figure 2. We note that only one set of each of the sizes has been sampled to compute the sketch (i.e., $k = 1$). Recall that the distance returned by Online-Common-Seed is based on finding common nodes in the sketches of the two query nodes. On the other hand, the lower bound algorithm Online-Bourgain is based on computing the L_∞ distance between distances of query nodes to sampled sets.

Consider the plots for undirected distances first. Notice in Figure 1 that all the three quartiles produced by Online-Common-Seed are fairly close to $x = y$. In fact, even around the maximum true distance of 15, we obtain an upper bound of about 18 even at the 75th percentile. This suggests a 1.2-approximation ratio. Considering that all the previous approaches can only guarantee a $O(\log n)$-factor approximation (sometimes with large constants) and do not scale well, this is a very good approximation. In the same figure, we see that the lower bound produced by Online-Bourgain is also reasonably good (steadily increasing). However, it is noisier than the upper bound, and with a weaker guarantee, of about 15/7 which is 2.14-approximate.

In the case of directed distances (again with just one set sampled for each of the $\log n$ sizes), Figure 2 illustrates a large gap between the estimates produced by Online-Common-Seed and Online-Bourgain. While Online-Common-Seed produces an extremely good 1.05-approximate estimate of the true distance, Online-Bourgain performs rather poorly. This is due to the fact that the number of sampled sets was not enough to capture a directed path between many pairs of nodes. Indeed, it is very difficult to capture a directed path through a sampled seed set when there may be very few directed paths. In the following set of experiments, we study the effect of larger sketches on the performance of both algorithms.

Using larger sketches to improve the bounds

We can improve the large gap in the directed distance estimation by additional sampling. We construct $k = 20$ sets of

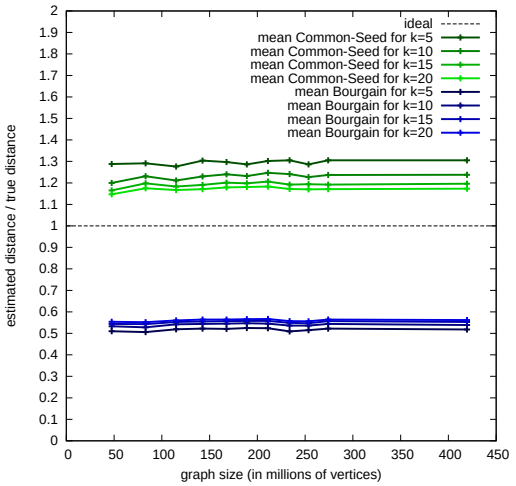

Figure 10: Effect of graph size on the estimated undirected distance

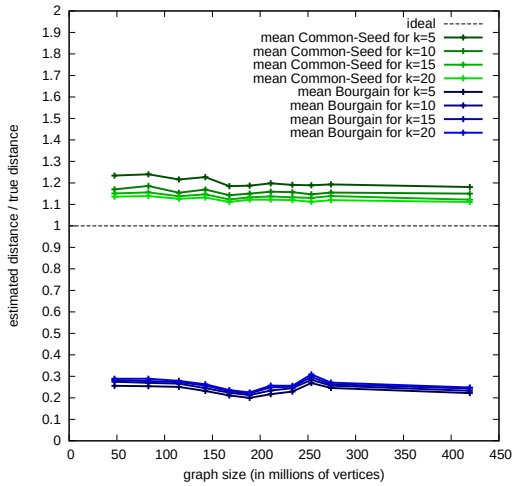

Figure 11: Effect of graph size in the estimated directed distance

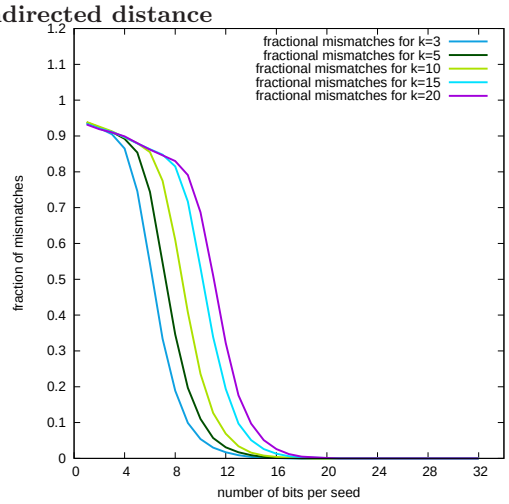

Figure 12: Effect of the number of bits per seed on the number of false positives - undirected case

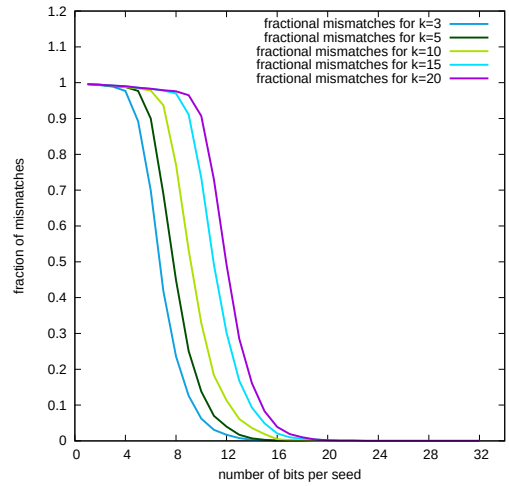

Figure 13: Effect of the number of bits per seed on the number of false positives - directed case

each of the $\log n$ different sizes, and use them to construct the sketch of every node. The hope is to capture many more directed paths between query nodes and the sets (which are available in the sketch) since the total number of sets in the sketch has gone up from $\Theta(\log n)$ to $\Theta(k \log n)$. This in turn yields better estimates of the directed distance between the two query nodes. The bounds computed using a larger number of sampled sets are plotted in Figure 3 and 4. Instead of plotting the actual estimated distance, we plot the ratio of estimated distance to true distance on the y-axis. Thus, this value will be above the ideal $y = 1$ line for the upper bound and below this line for the corresponding lower bound.

The sampling turns out to dramatically improve the upper bounds. All three – median, 75th percentile, and 25th percentile approach $y = 1$ as the number of samples increase. Note that the upper bound estimate reduces from a 1.6-approximate value for $k = 1$ to a 1.25-approximate value for $k = 20$. This shows that the algorithm Online-Common-Seed is successful in computing directed distance estimates to a high accuracy as well, provided sufficient sampling is done in the offline sketching phase. On the other hand, the lower bound from Bourgain's algorithm still does not quite match the performance of Online-Common-Seed though it does produce a 1.85-approximate distance value for $k = 20$.

In the above experiment, we compute the quartile values for the ratios over all distances. To visualize the effectiveness of our algorithms for any given distance, we plot the median value of the ratio of the estimated distance to the actual distance for the undirected and directed cases in Figures 5 and 6 respectively. Further, we consider three values of k, 1, 10, and 20, to also illustrate the effect of the number of samples. As these figures show, the accuracy of the upper bound increases as the true distance increases. This is explained by the low likelihood of two vertices having a common seed in one of the smaller seed sets. This trend is not observed for the lower bound where, in fact, the ratio for the lower bound approaches a value close to 0 for large distances. The second observation is that the number of samples also helps in improving the distance estimates in both directions, albeit more in the case of the upper bound.

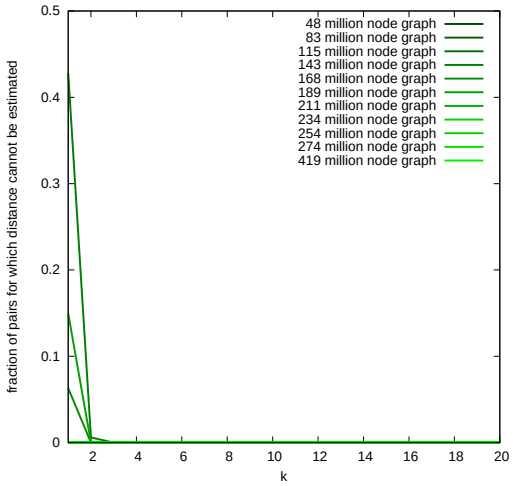

Figure 9: Fraction of non-covered (u, v) pairs for different k using Online-Common-Seed

We note that a sample size of $k = 10$ gives a good approximation for true distances above 10. This suggests that we can achieve good performance from Online-Common-Seed using a reasonably small number of samples.

To illustrate the spread of the error in distance estimation, we compute the distribution of the estimated distances for a given distance. Figures 7 and 8 show the distribution of the error for the lower and upper bounds where the distance $d = 10$ is chosen for the undirected case and distance $d = 50$ is chosen for the directed case. We chose these values of distance from the middle of the range for each case. Again, we run the experiment for three values of k: 1, 10, and 20. As we can see from the figures, there is a sharp concentration of around 0.5 (2-approximate) for the lower bound that does not change much with k. Further, the value of k does not affect the the sharpness of the concentration. On the other hand, for values produced by Online-Common-Seed, we observe two trends. First, the sharpness of the concentration increases with k. Second, the concentration shifts toward the ideal value of 1 as k increases.

In the case of directed distances, Online-Common-Seed produces a much better distribution with a concentration around 1.05 while Online-Bourgain produces a concentration close to 0.1. We see the concentrations shift slightly toward the ideal value of 1.0 (from both directions) as the value k increases showing the effectiveness of using more number of samples.

In the earlier experiments, we ignored all pairs between which our algorithms could not find a path. In the undirected case there are no such pairs as our graph is connected. To show that there are not many such entries in the directed case, we compute for different values of k the fraction of sampled pairs for which the algorithms fail to estimate a finite distance. Figure 9 illustrates the trend in the fraction of uncovered pairs as the number of samples increases. This value quickly drops to 0 even as k increases slightly above 2. This trend is observed for almost all the graphs. This result combined with that in Figure 4 shows that our algorithm finds a good approximation to the actual directed distance between *any* given pair of vertices.

Effect of graph size

In another set of experiments, we varied the graph sizes by choosing different prefixes of the breadth-first-search crawl and estimated both directed and undirected distances for a sampled set of vertex pairs. Specifically, we considered graphs of sizes between 47 million and 419 million nodes. The corresponding number of edges varied from about 171 million to 2.3 billion. Figures 10 and 11 illustrates the effect of graph size on the estimation of undirected and directed distances respectively. While there is no noticeable effect on the estimated distance in the undirected case, Online-Common-Seed tends to do better with larger graph sizes. One explanation for this phenomenon could be that there are likely to be more paths between nodes in a larger graph and hence can be captured by the seed sets.

Effect of seed size

So far we have used 32-bit unique node identifiers to represent the seeds in our sketch computation. We will now explore the effect of lossy compression of the seeds. We hash the 32-bit seed representation to a representation with fewer, say b, bits. Note that the seeds are not used in Online-Bourgain and hence the performance of this algorithm is not affected. In the algorithm Online-Common-Seed, we use the seeds in the sketch to identify the common seed in the sketches of a given pair of nodes. Hashing the seed ID into a representation with fewer bits might produce false positives of common seeds in the process. In fact, the probability of two seeds hashing to the same value is $1/2^b$ and since the number of distinct pairs of seeds across the two sketches grows quadratically in k, the total probability of a false positive is proportional to $\frac{k^2}{2^b}$. Note that because of such false positives, the algorithm may not necessarily return an upper bound of the true distance. Not surprisingly, we observe that for small values of k and b, we obtain reasonable accuracy in the estimates of the upper bound, i.e., we see a tiny fraction of false positives. Figures 12 and 13 show the fractional mismatches for different values of b and k. In the undirected case, we need fewer bits to achieve the same accuracy compared to the directed case. However, as k increases, the number of bits required to reduce the false positives also increases as the probability of a false positive is proportional to k^2. We observe that for $k = 3$ and 12 bits per seed, the fraction of mismatches is almost negligible. Further, the error in the estimated distance for $k = 3$ is also small (see Figures 3 and 4). This sketch size (in bits) can be computed as $(s + 8)k \log n$, where the $\log n$ factor comes from the number of seed sets and s is the number of bits per seed. The additional 8 bits store the distance. Setting $k = 3$, $s = 12$ and the number of seed sets to 32, we get a sketch size of 240 bytes for undirected graphs and 480 bytes for directed graphs.

6. CONCLUSIONS

We present an algorithm for obtaining sketches that support efficient distance queries. The sketches are computed offline and distances between any pair of nodes can be estimated online by looking at their sketches. While there has been theoretical work for this problem, few of the approaches scale to web graphs. We present an algorithm that can approximate true distances and provide strong theoretical guarantees for the upper bound achieved by our algo-

rithm Online-Common-Seed. Furthermore, while all previously suggested algorithms estimated distances on undirected graphs, our techniques can be extended to estimate distances on directed graphs as well. We compare our algorithm with Bourgain's well-known algorithm based on embedding graphs into low-dimensional metric spaces. The lower bound given by Bourgain's algorithm turns out to be significantly weaker. It is unable to predict directed results with any precision. For undirected graphs, while it is unable to match our upper-bound algorithm in terms of accuracy, it does give a reasonably good prediction.

We conduct extensive experiments of all the algorithms proposed in this paper and compare it with true distances for sampled pairs of nodes. The experiments are run on a crawl of a large web graph. As far as we are aware, this is the first practical work on distance oracles that has been adapted to real data at this scale. We plan to run these algorithms on other large graphs such as an Instant Messenger graph.

Acknowledgements

We would like to thank Ittai Abraham for helping us prove stronger theoretical guarantees on our distance estimation algorithm.

7. REFERENCES

[1] Yair Bartal. On approximating arbitrary metrices by tree metrics. In *STOC*, pages 161–168, 1998.

[2] Surender Baswana. Streaming algorithm for graph spanners - single pass and constant processing time per edge. *Inf. Process. Lett.*, 106(3):110–114, 2008.

[3] Jean Bourgain. On Lipschitz embeddings of finite metric spaces in hilbert space. *Israel Journal of Mathematics*, 52(1-2):46–52, 1985.

[4] Andrei Z. Broder. Identifying and filtering near-duplicate documents. In *CPM*, pages 1–10, 2000.

[5] Andrei Z. Broder, Steven C. Glassman, Mark S. Manasse, and Geoffrey Zweig. Syntactic clustering of the web. *Computer Networks*, 29(8):1157–1166, 1997.

[6] Edith Cohen, Eran Halperin, Haim Kaplan, and Uri Zwick. Reachability and distance queries via 2-hop labels. In *SODA*, pages 937–946, 2002.

[7] Reuven Cohen, Pierre Fraigniaud, David Ilcinkas, Amos Korman, and David Peleg. Labeling schemes for tree representation. *Algorithmica*, 53(1):1–15, 2009.

[8] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. Introduction to algorithms. *MIT Press and McGraw-Hill*, 24.3:595–601, 2001.

[9] Camil Demetrescu, Andrew V. Goldberg, and David S. Johnson. Implementation challenge for shortest paths. In *Encyclopedia of Algorithms*. 2008.

[10] Wei Dong, Moses Charikar, and Kai Li. Asymmetric distance estimation with sketches for similarity search in high-dimensional spaces. In *SIGIR*, pages 123–130, 2008.

[11] Jittat Fakcharoenphol, Satish Rao, and Kunal Talwar. A tight bound on approximating arbitrary metrics by tree metrics. *J. Comp. Syst. Sci.*, 69(3):485–497, 2004.

[12] Joan Feigenbaum, Sampath Kannan, Andrew McGregor, Siddharth Suri, and Jian Zhang. Graph distances in the streaming model: the value of space. In *SODA*, pages 745–754, 2005.

[13] François Fouss, Alain Pirotte, Jean-Michel Renders, and Marco Saerens. Random-walk computation of similarities between nodes of a graph with application to collaborative recommendation. *IEEE Trans. Knowl. Data Eng.*, 19(3):355–369, 2007.

[14] Cyril Gavoille, David Peleg, Stéphane Pérennes, and Ran Raz. Distance labeling in graphs. *J. Algorithms*, 53(1):85–112, 2004.

[15] Andrew V. Goldberg. Point-to-point shortest path algorithms with preprocessing. In *SOFSEM (1)*, pages 88–102, 2007.

[16] Michal Katz, Nir A. Katz, Amos Korman, and David Peleg. Labeling schemes for flow and connectivity. *SIAM J. Comput.*, 34(1):23–40, 2004.

[17] Jiri Matousek. On the distortion required for embedding finite metric spaces into normed spaces. *Israel Journal of Mathematics*, 93(1):333–344, 1996.

[18] Marc Najork. The scalable hyperlink store. In *Hypertext*, pages 89–98, 2009.

[19] Mikkel Thorup and Uri Zwick. Approximate distance oracles. *J. ACM*, 52(1):1–24, 2005.

[20] Luh Yen, Marco Saerens, Amin Mantrach, and Masashi Shimbo. A family of dissimilarity measures between nodes generalizing both the shortest-path and the commute-time distances. In *KDD*, pages 785–793, 2008.