Estimating Pairwise Distances in Large Graphs

Maria Christoforaki
NYU Polytechnic School of Engineering
Brooklyn, NY
mc3563@nyu.edu

Torsten Suel
NYU Polytechnic School of Engineering
Brooklyn, NY
torsten.suel@nyu.edu

***Abstract*—Point-to-point distance estimation in large scale graphs is a fundamental and well studied problem with applications in many areas such as Social Search. Previous work has focused on selecting an appropriate subset of vertices as landmarks, aiming to derive distance upper or lower bounds that are as tight as possible. In order to compute a distance bound between two vertices, the proposed methods apply triangle inequalities on top of the precomputed distances between each of these vertices and the landmarks, and then use the tightest one.**

In this work we take a fresh look at this setting and approach it as a learning problem. As features, we use structural attributes of the vertices involved as well as the bounds described above, and we learn a function that predicts the distance between a source and a destination vertex. We conduct an extensive experimental evaluation on a variety of real-world graphs and show that the average relative prediction error of our proposed methods significantly outperforms state-of-the-art landmark-based estimates. Our method is particularily efficient when the available space is very limited.

I. Introduction

The design of efficient graph algorithms has received a lot of attention in computer science, and algorithms for computing shortest paths have played a central role within this literature. Shortest paths have important applications in many areas, including social networks, where the point-to-point distance between two nodes is of significant interest. Users with common interests are more likely to be connected with each other in social networks, and thus the distance between two people in a social graph is considered an important signal regarding their similarity and the chance that they might know each other. Similarly, in a collaboration network, the distance between two participants can be a measure of how relevant one's work is to that of the other.

Given the importance of distances as strong signals of similarity and relevance, a lot of recent work has looked at how to use graph distances in various applications; see, e.g., [1], [2], [3], [4] for a small sample. Many of these applications, including content search on social networks, and social question-answering, are instances of *social search* [3]. Social search personalizes its results by also considering the social graph of the user who issued the query; thus, content that has been generated or recommended by users that are "closer" to the issuing user in the social graph receives higher priority. Traditional web search can also benefit from similar techniques, where the "context" within which a search query is initiated can be used to refine the results. Thus, a context-sensitive search engine could give higher priority to pages that are closer, in the web graph, to the page that the user is currently visiting [1].

One obstacle that all above scenarios face is that, given a query involving a pair of nodes, the point-to-point distance between them needs to be computed in milliseconds or even less. One way to ensure a quick response is to store all pairwise distances in main memory, but this results in a space requirement that is quadratic in the number of nodes, which is not feasible in many scenarios. On the other hand, running Breadth First Search or Dijkstras algorithm during query time is usually much too slow. A common approach is to preprocess the graph and store a limited amount of encoded information that can be accessed during query time to quickly estimate the distance between two nodes.

Most previous work has followed this approach by focusing on methods that return upper or lower bounds for the distance between two nodes based on small sets of nodes called *landmarks* [5], [6], [2], [7], [8]. In the pre-processing steps, these methods choose an appropriate set of landmarks, and then compute and store distances from every node to each (or some) of these landmarks. Given a query consisting of two nodes whose distance needs to be computed, the distances between the two nodes and each landmark are retrieved. Applying the triangle inequality yields upper bound and lower bounds for the distance between the two nodes. Although these methods often provide reasonably accurate estimates, there is still a lot of room for improvement.

In this work we revisit the problem of estimating pairwise distances, and approach it as a learning problem. We find that for small-world networks, structural properties of nodes such as centrality metrics hold some information about their "position" in the graph and can therefore be used as strong signals for predicting pairwise distances. Moreover, we propose to treat the bounds computed by landmarks as informative signals rather than direct estimates for the actual distance between two nodes. Our goal is to learn a function that provides a better estimate of the pairwise distance than the strict bounds obtained by previous work. Altogether, we propose three methods for estimating distances that are simple, easy to implement, and highly efficient: one based on landmark bounds, one based centrality metrics, and one based on a combination of these. We then perform a detailed evaluation and show that our new approaches are superior to the existing state-of-the-art, particularily when space is very limited.

In the remainder of the paper, we first review previous research on pairwise distance computation in graphs. In Section III, we provide some notations and definitions. Section IV describes our new methods for distance estimation, while our experimental evaluation is presented in Section V. Finally, we provide some discussion and concluding remarks.

II. Background and Related Work

A. Exact Pairwise Distance Computation

The most commonly used algorithm for computing all the shortest paths from a single source is Breadth First Search (BFS). This algorithm is applicable to unweighted graphs $G = (V, E)$ and has $O(|E| + |V|)$ time complexity. For weighted graphs Dijkstra's algorithm [9] is used instead and its time complexity is $O(|E|+|V|\log|V|)$ when implemented with a Fibonacci heap. Bidirectional Dijkstra runs two simultaneous searches, from the source and the target vertices, and stops when the two meet. This decreases the search space significantly in many cases and therefore is often a more efficient solution for point-to-point shortest path (PPSP) length computation.

For PPSP length computation, the state-of-the-art solutions are the ALT algorithms [10], [11], [12] which combine bidirectional Dijkstra with A^* search. ALT algorithms prune graph paths during run-time and reduce the search space of point-to-point distance queries, leading to improved efficiency compared to Dijkstra's algorithm. Most of the literature [10], [11], [12], [13], [14] related to ALT algorithms has focused on spatial networks, such as road networks. In fact, these algorithms can compute the exact values of pairwise distances relatively fast when the underlying graph satisfies certain useful structural properties, such as near planarity, low degree, and the presence of hierarchy, which holds for road networks. However, the running time of these algorithms increases dramatically when these properties are not satisfied, and thus they are not practical when it comes to interesting graph structures such as social networks, web graphs, and co-authorship networks, which do not exhibit such properties.

Another exact distance computation for method directed graphs was proposed by Cohen et al. [15]. It is a 2-hop labeling method where each vertex v stores a list of "intermediate" vertices $OUT(v)$ that it can reach, along with their distances. It also stores a corresponding $IN(v)$ list for the vertices that can be reached by v. During query time the distance of a pair of vertices (u, v) is computed by simply retrieving the $OUT(u)$ and $IN(v)$ lists and returning the minimum distance $d(u, l) + d(l, v)$ for all vertices l in the intersection of the two lists. The authors of this work also primarily utilize the highway structure of networks to speed up online search, which makes it usuitable for networks with a different structure such as social networks. Besides that, the minimum storage requirements are conjectured to be $O(nm^{1/2})$ for directed graphs, which is close to quadractic in the number of vertices and is prohibitive for large graphs. Jin et al. [16] propose a similar approach called Highway Centric Labeling to compute pairwise distances in sparse graphs. They show a significant reduction in query time compared to Cohen et al. [15]. However, their results show that the average degree significantly affects the performance. On a graph of $5K$ vertices, increasing the average degree from 1.2 to 2 causes a ten times slower query processing time (from 115.2 milliseconds to 9.5 seconds). These results render this approach inappropriate for graphs like social networks that have average degrees ranging from about 10 to more than 100.

Fu et al. [17] propose IS-LABLE, which also considers a vertex hierarchy to compute exact distances. However, the results show that for graphs of $100K$ and $200K$ vertices the index size reaches 7 and 8 GB, respectively, and the corresponding query processing times are 6 and 28 milliseconds. We are studying settings where the available space is very constrained and and the query processing time has to be in the order of microseconds even for graphs of millions of vertices.

To the best of our knowledge, Akiba et al. [18] is currently the fastest method for computing exact distances for complex graphs like social networks. They propose a 2-hop labeling approach. In particular, they suggest a preprocessing step which results in a landmark set $S(v)$ for every vertex v, such that for every pair of vertices (u, v) in the graph there is at least one vertex $w \in S(u) \bigcap S(v)$ that lies on the shortest path between u and v. Their results show very fast query times compared to previously proposed methods ($15\mu s$ for complex graphs of one million vertices) but require extremely large index structures ($12GB$ for complex graphs of just one million vertices).

B. Pairwise Distance Estimation

A significant amount of work, reviewed next, has focused on designing efficient solutions that compute *estimates* for pairwise distances.

Theoretical Results for Pairwise Distance Estimation. Thorup and Zwick [19] propose a multilevel sampling approach followed by a per-vertex landmark selection process. Given an integer $k \geq 1$, and a graph $G = (V, E)$, the proposed solution generates a sequence of $k + 1$ samples starting from V, and then iteratively removes each vertex of that sample with probability $1 - n^{-\frac{1}{k}}$. After each iteration, this process yields a sample which is a subset of the sample that the iteration started with. For each of these subsets, every vertex in the graph chooses the one closest to it and stores its id along with its distance to it. Given a point-to-point distance query, the algorithm tests whether the two endpoints have a common landmark, starting from the closest ones, and returns the distance estimate based on the first match. Thorup and Zwick prove that any graph can produce a node sketch of size $O(kn^{1+(1/k)})$ for any constant $k \geq 1$, that using these sketches any point-to-point query can be answered in $O(k)$ time, and that the estimate is a $2k - 1$ approximation of the real distance. Setting $k = 1$ gives the exact distances for every point-to-point query, but requires $O(n^2)$ space consumption. Increasing k to 2 requires $O(2n^{\frac{3}{2}})$ space, which is prohibitive for large graphs. This gives a 3-approximation of the distance which is too large especially for applications where the graphs have a small diameter.

Another line of work has proposed use of embedding methods [20], [21], [22], [23], [24], [25]. These methods first map high-dimensional objects (nodes in a graph) to lower dimensions, and define an appropriate and efficiently computable distance function in this lower-dimensional space; then this function is shown to provide distance estimates that are provably close to the actual distance. Another line of work makes use of sparse subgraphs of the initial graph called spanner graphs [26], [27], [19]. Both these approaches provide theoretical bounds for the space requirements of the proposed solutions and the extent to which the actual distances are approximated. However, most of the solutions in this literature are rather complicated in terms of implementation, and no empirical evaluation is provided. In this work, our heuristics also map each vertex to a lower-dimensional space and derive a distance function in this space. Our methods do not have theoretical guarantees, but are simple to implement, and we show that they work well in practice in terms of both approximation error and space consumption.

Heuristics for Pairwise Distance Estimation.
In tandem with this theoretical work, several studies have also approached this problem from an empirical perspective [5], [6], [2], [7], [8]. The solutions proposed by these studies choose a subset of the graph nodes, called *landmarks*, and pre-compute the shortest-path lengths between all the graph nodes and some or all of these landmarks. Then, given a query, these pre-computed distances are combined using the triangle inequality in order to compute upper bounds for the exact distance. These results can be divided into two main categories, based on the use of these landmarks. The *global landmark* approaches [5], [6], [2] pre-compute and store the distances of every vertex to every landmark. Several different global landmark selection strategies have been proposed: Vieira et al. [5], to the best of our knowledge, is the first paper to study distance estimation in social networks. The authors study the effects of random global landmark selection. Potamias et al. [2] suggest a number of more sophisticated landmark selection techniques based on centrality properties of the nodes; intuitively, their goal is to select a set of landmarks that are likely to be found on many shortest paths between nodes. They show that their landmark selection strategy can significantly reduce the estimation errors compared to random selection, given the same storage constraints.

Qiao et al. [6] point out that, although centrality metrics are good signals for the "appropriateness" of a vertex as a landmark, when greedily selecting the landmarks using such metrics, the marginal contribution of each new landmark is often small. They suggest an alternative greedy algorithm measuring the graph's "coverage" by the chosen landmarks, as well as an algorithm which first partitions the graph and then chooses good landmarks for each partition.

Another set of literature has focused on methods for estimating distance lower bounds [10], [11], [12], [13], [2], [8]. Lower bounds on distances are especially useful for pruning the search space in graph search algorithms. However, it has been observed that lower bounds using landmarks and the triangle inequality are much worse predictors of the actual distance than upper bounds [2]. In this work, we leverage upper and lower bounds computed by landmarks to derive one of our proposed distance functions.

Alternatively, *local landmark* approaches [8], [19], [28], store only a subset of the selected vertex-landmark distances for each vertex. This is motivated by the fact that not all global landmarks are useful for every vertex in the graph. It is therefore more efficient if each vertex stores only those landmark distances that are more likely to give tight estimates for this particular vertex. Das Sarma et al. [8] simplifies the work of Thorup and Zwick [19], allowing for an easier implementation while retaining the theoretical guarantees. Furthermore, Das Sarma et al. empirically evaluated their algorithm and show that in practice the estimation errors are usually much better than the theoretical guarantees. In our empirical evaluation we found that the local landmark method proposed by Das Sarma et al. gives slightly tighter bounds only for distance-1 queries, compared to global landmark methods. In contrast, it has a significantly (2 to 3 times) larger expected error for other point-to-point distance queries, compared to the best global landmark-based methods given the same space consumption.

Another recent paper proposing a local landmark scheme is the work of Qiao et al. [28]. The authors of this work extend the state-of-the-art global landmark methods by introducing an additional index for every landmark. The role of this structure during query processing of a vertex pair u, v is to find the least common ancestor of u and v in the SPT rooted at each landmark, and use the path passing through the least common ancestor of u, v as a distance upper bound. They propose two alternative methods, *LLS* and *LS*. *LS* results in very accurate estimates compared to state-of-the-art landmark-based approaches. It also outperforms their alternative method *LLS* in terms of estimation accuracy. However, to achieve this small error, it requires significant amounts of space to store its indexes, as well as significant time to process a query. For the Youtube graph, for instance, it requires 2 to 4 milliseconds for a 0.3% error. If the available space is large, the exact distance estimation methods proposed in [18] are more appropriate solutions as they are faster and have no error. On the other hand, *LLS* is faster and requires less space, but is less accurate than our proposed method.

C. Other Related Work

A relevant and slightly more complex problem that has received some attention is that of shortest path computation [7], [29]. In [7], [29], the authors propose methods for efficiently computing the set of edges composing a shortest path between two vertices, rather than just computing the length of that path. It is also worth mentioning that there is some somewhat relevant work from the database community on query processing [30], [31], [32], [33] in graphs. These papers mostly focus on reachability queries rather than shortest path distance computations.

In a slightly different context, in [34] the authors propose a method to learn a structure-preserving distance metric for graphs with respect to different connectivity algorithms. Their goal in this work is to reconstruct the graph using only a small amount of precomputed information for each vertex. This work focuses on the problem of predicting whether an edge exists or not, rather than on computing distances between arbitrary nodes.

III. Preliminaries

In this section we provide some notation and define the problem that we study.

A. Problem Definition

Let $G = (V, E)$ denote an undirected and unweighted graph, with $n = |V|$ vertices and $m = |E|$ edges. Given any pair of vertices $u, v \in V$, let $P_{u,v}$ be a path connecting u and v; formally, $P_{u,v} = \{u, a_1, a_2, \ldots, a_{l-1}, v\}$, where $a_i \in V$ for all $i \in \{1, 2, \ldots, l-1\}$ and $(u, a_1), (a_1, a_2), \ldots, (a_{l-1}, v) \in E$. The point-to-point distance $d(u, v)$ between the two vertices u, v is the length of the shortest path among all paths that connect u and v, i.e., $d(u, v) = \min_{P_{u,v}} |P_{u,v}|$. Finally, the diameter $\Delta = \max_{u,v \in V} d(u,v)$ of a graph G is the longest shortest path connecting any two vertices in the graph G.

In this work, we focus on the problem of designing a solution for serving real-time point-to-point distance queries. As discussed above, computing the exact distance between two arbitrary vertices u, v online is time consuming, and hence impractical for many real-time applications. On the other hand, pre-computing and storing all the pairwise distances in a look-up data structure requires $O(n^2)$ space, which is prohibitive for graphs of millions of nodes. Therefore, our goal is to provide an "approximate" solution that is fast without requiring too much space. In particular, we aim to implement a function $f(u,v)$ that, given two vertices u and v, computes an estimate of their distance $d(u,v)$ with a low relative error

$$\frac{|d(u,v) - f(u,v)|}{d(u,v)}.$$

B. Vertex Centrality Attributes

In order to define our solutions, we will be using vertex centrality attributes that provide some measure of how well-connected a vertex is with the rest of the graph.

Vertex betweenness. The betweenness $B(v)$ of a vertex v provides a measure of how central the position of this vertex is in the graph. It is defined as the ratio of the number of shortest paths that go through the vertex and the total number of shortest paths. Formally, let $P(s,t)$ be the set of shortest paths between two vertices s, t, and let $P = \bigcup_{s,t} P(s,t)$ be the union of all these sets. Then,

$$B(v) = \frac{|\{p \in P \mid v \in p\}|}{|P|}.$$

Vertex closeness. The closeness-centrality $C(v)$ of a node v aims to quantify how close this vertex is to every other vertex of the graph. In particular, it is the inverse of the sum of the distances of v to every other vertex u:

$$C(v) = \frac{1}{\sum\limits_{u \in V} d(u,v)}$$

Vertex n$^{\text{th}}$ degree. The n^{th} degree $D_n(v)$ of a vertex v is the natural generalization of the commonly used first degree, and it measures how many vertices of the graph are at most n hops away from v, that is,

$$D_n(v) = |\{u \in V \mid d(u,v) \leq n\}|.$$

C. Landmarks

A set of landmarks is an appropriately selected subset of vertices $L \subset V$ that can be used in order to derive upper and lower bounds regarding the distance between any two vertices via the triangle inequality. In particular, given a landmark $l \in L$ and any two vertices $u, v \in V$, applying the triangle inequality implies that

$$|d(u,l) - d(l,v)| \;\leq\; d(u,v) \;\leq\; d(u,l) + d(l,v).$$

Hence, a landmark set L defines a range $[B_L^-(u,v), B_L^+(u,v)]$ within which the actual distance $d(u,v)$ needs to lie, where

$$B_L^-(u,v) = \max_{l \in L}\{|d(u,l) - d(l,v)|\} \tag{1}$$

is the maximum lower bound over all landmarks in L, and

$$B_L^+(u,v) = \min_{l \in L}\{d(u,l) + d(l,v)\} \tag{2}$$

is the minimum upper bound.

IV. Learning to predict distances

Our the key objective is to generate highly accurate distance estimates based on minimal information stored about the graph. Ideally, our goal would be to store only a fixed amount of information per node, leading to space requirements that grow linearly, instead of quadratically with the number of nodes.

Formally, one can think of such a solution as a mapping $Q_1 : V \rightarrow \mathbb{R}^\alpha$ that represents each node in an α-dimensional space, for some constant α. This mapping can be computed during pre-processing and then, during processing of a query involving a pair of nodes (u,v), the solution combines the representations $q_u = Q_1(u)$ and $q_v = Q_1(v)$ of the two nodes in order to generate a distance estimate.

The state-of-the art solutions for the problem of point-to-point distance estimation [5], [6], [2], [10], [7], [8] are in fact instances of this approach: during pre-processing time these algorithms choose a subset of nodes L to serve as landmarks, and the distance of every node from every landmark is computed. The representation q_u of a node u then corresponds to the vector of distances $d(u,l)$ of u from each landmark $l \in L$, i.e., $\alpha = |L|$ and

$$q_u = [d(u,l_1), d(u,l_2), \ldots, d(u,l_{|L|})]^T \tag{3}$$

Then, given a point-to-point query for a pair of nodes (u,v), these solutions respond with either $B_L^-(u,v)$ or $B_L^+(u,v)$, which they compute during processing time using the information from the representations q_u, q_v and Equations (1) and (2).

Although these solutions satisfy the desired space constraints and give guaranteed upper and lower bounds for the distance, they do not necessarily lead to a small error. In light of this observation we, revisit the problem, as a learning problem aiming to derive a distance estimation function that outputs an improved estimate. The functions that we propose are linear combinations of a set of signals including structural attributes of the nodes involved as well as the upper and lower bounds induced by the landmark-based methods.

A. Learning Distances using Vertex Properties

Our first approach uses pre-processing in order to compute some structural attributes, such as centrality metrics, of each node in the graph. The values of these attributes for each node u then form the representation q_u of the node, which provides us with some information about the node's "position" in the graph. In particular, centrality attributes like the degree and the closeness of a node are directly related to its expected distance from a random node in the graph, and thus they provide very useful structural signals, especially for dense graphs with small diameter, and a skewed degree distribution, such as social networks. On the other hand, as we also point out in Section V, this may not be true for some types of graphs like grids or road networks, where the distances are larger, the graphs are sparser, and the distribution of the degrees more uniform. The specific attributes that we compute for each node u are its closeness-centrality, its betweenness-centrality, and its 1st, 2nd, and 3st degree. Thus we have:

$$q_u = [B(u), C(u), D_1(u), D_2(u), D_3(u)]^T. \quad (4)$$

The 1st, 2nd, and 3nd degree are easily computed. However, computing the exact closeness-centrality and betweenness-centrality can be very time consuming for large graphs. Thus, we instead compute an approximation of closeness and betweenness centralities that is commonly used in practice: we first sample a subset of the graph's nodes uniformly at random, and then run BFS from the subset to the rest of the graph to compute the distances and calculate closeness. During this process we also apply the algorithm of [35] to compute the betweenness. Note that, unlike the existing algorithms mentioned above, where the value of α corresponds to the number of landmarks used and can take large values, the value of α in this approach is fixed to 5.

Now, when a distance query for a node-pair (u, v) arrives, we combine the representations q_u and q_v of the two nodes into a representation $q_{u,v}$ of the node-pair using a mapping $Q_2 : \mathbb{R}^\alpha \times \mathbb{R}^\alpha \to \mathbb{R}^\beta$. Then we use a trained function $f_v(\cdot)$ which, given $q_{u,v}$, outputs an estimate for $d(u, v)$. In order to train $f_v(\cdot)$, each element of the β-dimensional vector corresponding to the representation $q_{u,v}$ is used as a feature, and weights are computed for each one of these features using linear regression.

The particular representation $q_{u,v}$ that we use merges the corresponding attributes of q_u and q_v, giving higher priority to the one with higher value. That is, the first element of $q_{u,v}$ is $\max\{B(u), B(v)\}$, the second one is $\min\{B(u), B(v)\}$, the third one is $\max\{C(u), C(v)\}$, and so on. The reasoning behind this choice is that, given two vertices u and x such that $q_u = q_x$, we would like to ensure that $q_{u,v} = q_{x,v}$ for any other vertex v. Since the distance function is a symmetric one, violating the property described above would hinder the learning algorithm's ability to generalize.

B. Learning Distances using Landmarks

The above function using structural properties of vertices as features is based on signals that the two vertices give regarding their "position" in the graph. Another option is to use the distance bounds computed by landmarks as such signals, thus using the existing landmark-based distance-bounding methods as subroutines. Based on this idea, we propose a second family of distance estimation functions that begin with the same representation q_u for each vertex u defined in (3), but then we use linear regression in order to learn how to better combine these representations into distance estimates.

In choosing a representation $q_{u,v}$ that combines the information in q_u and q_v, our goal was to include useful signals regarding the true value of the distance $d(u, v)$. Hence, one choice would be to let $q_{u,v}$ be the vector of the $|L|$ upper bounds implied by the landmark distances in q_u and q_v:

$$q_{u,v} = [d(u, l_1) + d(l_1, v), \dots, d(u, l_{|L|}) + d(l_{|L|}, v)]^T. \quad (5)$$

Applying linear regression in order to train a function $f(\cdot)$ that aims to predict the distance $d(u, v)$ given $q_{u,v}$ amounts to treating each one of the β elements of $q_{u,v}$ as features, and learning a weight value for each one of them. This means that, if we were to use the representation of (5), then the training process would assign higher weights to landmarks that on average (across all vertex pairs in the training set) give tighter distance upper bounds, and smaller weights to landmarks that give looser distance bounds. However, this approach would be problematic since in general, no single landmark is expected to provide good predictions for most of the pairs. Instead, different landmarks provide good predictions for different pairs of nodes. In order to avoid these issues we instead let $q_{u,v}$ be the vector of *ordered* upper bound values from smallest to largest. As a result, the weights learned correspond to a profile of decreasingly tight bounds instead of specific landmarks. In what follows, we will refer to the distance prediction function that is trained using the ordered upper bounds representation as f_{ub}.

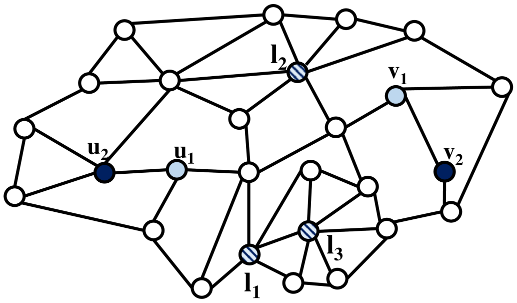

Fig. 1: Example for landmark distance bounds.

Note that all the previous distance prediction algorithms used only the tightest upper or lower bound provided by the chosen landmarks in order to respond to a given query. In contrast, the approach that we propose instead aims to extract useful information from more of the available bounds. For example, consider the graph of Figure 1, which has three landmark vertices $L = \{l_1, l_2, l_3\}$. The distance of vertex pair (u_1, v_1) is $d(u_1, v_1) = 3$ whereas the distance of pair (u_2, v_2) is $d(u_2, v_2) = 5$. The unordered representation of the node-pair (u_1, v_1) is $[5, 5, 6]$ and that of node-pair (u_2, v_2) is $[7, 5, 7]$. A traditional distance upper-bounding method using these landmarks would output the tightest estimate for each pair, which in this example is 5 for both of the pairs. However, if we consider all bounds that the landmarks suggest, we may suspect that while we know that both pairs of vertices are at most 5 hops away from each other, u_1 is likely to be closer to v_1 than u_2 is to v_2. Based on this idea, we give all bounds computed by the landmark set as an input to our distance estimation function, which will learn the importance of the information that each bound carries.

In addition to the f_{ub} function, which is trained using the ordered upper bounds representation, we also study the performance of f_{lb}, which is trained using an ordered lower bounds representation, and $f_{ub,lb}$ which uses both upper and lower bounds. For f_{lb} we follow the same process with the difference that the lower bounds in the representation are sorted from largest to smallest. For $f_{ub,lb}$ the length of the representation $q_{u,v}$ is $2|L|$. The first half of the representation consists of non-decreasing upper bound values, and the second half of non-increasing lower bound values.

In order to gain a better understanding of how considering more than just the tightest bound improves the performance of our algorithms, we experiment with a threshold value t which, for a given set of landmarks L, controls how many of the tightest bounds are included in $q_{u,v}$. In other words, we study the effect of letting the input of f_{ub} have dimension t with $t < |L|$. Note that this is not the same as removing some of the landmarks, since each node pair's representation may disregard the bound provided by a different landmark.

Finally, we also study the effect of combining these landmark-based approaches with the ones based on vertex attributes into a single prediction function $f_{ub,lb,v}$. In particular, we combined the upper and lower bounds with the three centrality metrics into one representation .

V. Experimental Evaluation

A. Datasets

In order to measure the quality of the estimates produced using our approaches, we conducted an extensive experimental evaluation on the following 6 graphs of different size and structure, which we obtained from the Stanford Network Dataset Collection[1]:

- **Astro-Phys**: This graph represents an Astro-Physics collaboration network. Each node corresponds to an author and each edge covers a scientific collaboration between two authors who submitted at least one paper to the Astro Physics category.
- **DBLP**: This is also a co-authorship network where two nodes (authors of computer science papers) are connected by an edge if they have published at least one paper together.
- **P2P-Gnutella:** This graph represents a snapshot of the Gnutella peer-to-peer file sharing network. Each host is represented by a node and each connection between Gnutella hosts is represented by an edge.
- **Orkut**: Orkut is a free online social network. Each node represents a user of the network and each edge a friendship between the two users.
- **Youtube**: The Youtube video-sharing web site includes a social network where users friend each other. As in Orkut, each user is represented by a node and a friendship by an edge.
- **CA-RoadNet**: This graph is the road network of California. Each node represents an intersection or an endpoint and each edge a road connecting intersections and endpoints.

Table I lists some structural properties of the graphs that we used in our experiments. The first column is the number of nodes, the second the number of edges, the third the diameter of the graph, and finally the fourth column is the average clustering coefficient of each graph. Note that all graphs are unweighted and undirected.

Graph Name	$\|V\|$	$\|E\|$	diam	avg. clus. coef.
Astro-Phys	$18K$	$1967K$	14	0.6306
DBLP	$317K$	$1M$	21	0.6324
P2P-Gnutella	$62.6K$	$148K$	11	0.0055
Orkut	$3M$	$117M$	10	0.1666
Youtube	$1.1M$	$3M$	20	0.0808
CA-RoadNet	$1.96M$	$5.5M$	849	0.0464

TABLE I: Graph properties

Figure 2 shows the distance distribution for every graph in the experiments, which appears to be close to a normal distribution for all six graphs. Note that the first five graphs have a much smaller range in the distance distribution. This is of course to be expected, since there is an underlying social network structure in all of them. On the other hand, the distances in the CA-RoadNet graph range from 1 to 750.

B. Training and Testing

In order to compare the quality of the methods that we propose with that of the existing landmark-based distance bounding methods, we selected a testing set T for each graph; T comprises one million vertex-pairs selected uniformly at random from all the vertex-pairs in the graph. We separated the testing set into ten parts of 100K vertex pairs each and performed 10-fold cross validation, i.e., each part was used for testing a distance prediction function which was trained using the remaining 9 parts. We found that after a certain training set size (around 100K), the quality of the resulting distance estimation function converges independently of the graph size (a more detailed review of our experiments and our results can be found in the full version of the paper [36]). In order to evaluate the performance of each distance prediction function f, we computed the average relative error of its estimates for all pairs in T:

$$\frac{1}{|T|}\sum_{(u,v)\in T}\frac{|d(u,v)-f(u,v)|}{d(u,v)}$$

C. Baselines

In order to compare the performance of our solutions to the state-of-the-art, we implemented all the existing landmark-based distance-bounding methods and run experiments with them on all six graphs for landmark-sets L ranging from $|L| = 1$ to $|L| = 500$. Then, for each graph we identified the most accurate out of all these solutions (the one with the smallest average relative error on that graph's testing set) and used it as a baseline for our experiments on that graph. In fact, for every graph, its best method performed better than the rest for every number of landmarks, which allowed us to make an unambiguous winner selection for our baseline. Note that for the rest of the paper we will refer to the the baseline of each graph as *Base*. In particular, *Betweenness* (selecting landmarks by decreasing betweenness) outperformed all other landmark selection methods for all graphs except Astro-Phys and CA-RoadNet. For Astro-Phys *Adaptive-Max-Degree* [6] was the winning method, and for CA-RoadNet the winner was *Optimized-Farthest* [11]. Our experiments also showed that for the five "social" graphs, the upper bounds yield significantly tighter estimates than the lower bounds, and therefore they are more precise distance estimators. On the other hand, the lower bounds for the CA-RoadNet graph provided better estimates than the upper bounds. Due to space constraints, the corresponding figures are in the full version of the paper [36].

[1] http://snap.stanford.edu/data/

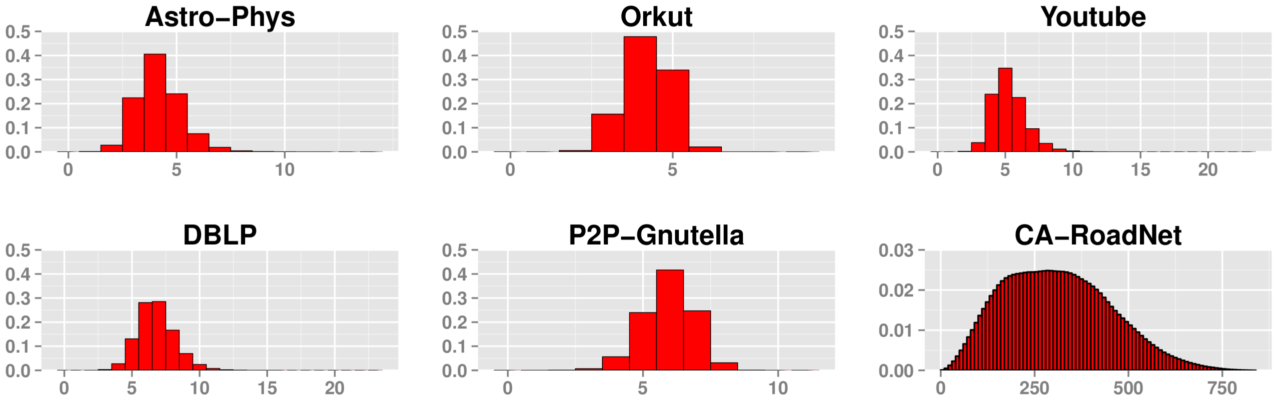

Fig. 2: Illustration of the distance distributions for the graphs used in the experimental evaluation.

D. Centrality Measures Approach

First, we studied the performance of the distance function f_v that was derived using the structural attributes of the vertices as features. We ran gradient descent with $\eta = 0.001$, which converged after about 10K iterations. Our initial experiments showed that the 3^{rd}-degree feature did not lead to coefficients with small p-values (its p-values varied from 0.13 to 0.24 for the six graphs), indicating that this feature was not providing useful information. Therefore, in order to improve the space consumption, we decided to remove this attribute from the representation q_u of the nodes, reducing the length of the representation from 5 to 4. On the other hand, $\min\{C(u), C(v)\}$ was the feature of $q_{u,v}$ with the highest coefficient and the smallest p-value (less than 10^{-7}) for all graphs, except roadNet-CA. The next highest weighted features, with p-values less than 10^{-6}, were $\min\{D_2(u), D_2(v)\}$ and $\max\{B(u), B(v)\}$.

This method needs to store the closeness, the betweenness, the 1st degree, and the 2nd degree for each node. In order to minimize the amount of space required, we stored the log values of the 1st and 2nd degree and normalized the closeness and betweenness values. Then we used a validation set and experimented with different quantization schemes for the feature values in order to reduce the number of bits to be stored per vertex without losing estimation accuracy. As a result, we reduced the space requirement to approximately 20 bits per vertex for all graphs except the roadNet-CA, which required 18 bits per vertex due to its smaller degrees.

On the other hand, the space requirement of the baseline approach is proportional to the number of landmarks being used. The addition of each new global landmark to L leads to an additional cost of a few bits for each vertex, since the vertex needs to store its distance from that landmark. In particular, each distance that a vertex needs to store per landmark can be encoded in $\log_2 \Delta$ bits, where Δ is the diameter of the graph. Hence, for the social graphs, the addition of each landmark costs 4 to 5 bits per vertex, whereas for the road network, the addition of each landmark costs 10 bits per node.

In Table II we compare the average relative error of f_v to that of the Base. In particular, the second column provides the average relative error of Base if it is restricted to using 20 bits per vertex, i.e., 4-5 landmarks (depending on the graph diameter)[2]. The last two columns of the table show the average relative error Base for two and three times the space consumption of f_v respectively. Note that the bits per vertex define the total size of the index for every graph. 20 bits per vertex for instance, result in index sizes of 44KB for Astro-Phys, 774KB for DBLP, 153 for P2P-Gnutella, 7.2MB for Orkut, 2.6M for Youtube, and 4.7MB for roadNet-CA, respectively. Moreover, the query processing in our method involves retrieving the vertex property values from arrays stored in main memory, and computing a linear combination with the learned weights.

Graph Name	f_v (20bpv)	Base$_{20\text{bpv}}$	Base$_{40\text{bpv}}$	Base$_{60\text{bpv}}$
Astro-Phys	0.091	0.261	0.201	0.173
DBLP	0.067	0.259	0.200	0.177
P2P-Gnutella	0.080	0.354	0.307	0.268
Orkut	0.107	0.201	0.166	0.152
Youtube	0.044	0.061	0.048	0.041
CA-RoadNet	0.423	0.383	0.247	0.208

TABLE II: Average relative error for f_v and Base.

We observe that for the five social graphs, f_v outperforms Base when the baseline is restricted to the same space consumption. In fact, even if we allow Base to use three times more space, f_v still either almost matches or even outperforms Base.

In contrast, f_v performs worse than the Base when it comes to the CA-RoadNet graph. This, is somewhat expected as the structure of the road network graph is very different than that of the social networks in terms of both distance distribution and degree distribution. The centrality metrics are therefore not able to generalize the graph structure as in the other graphs.

In addition to the very positive results of Table II, our experiments also indicate that the extent to which f_v outperforms Base in social graphs is even more pronounced when it comes to point-to-point distance queries whose distance is small (details deferred to [36]). This is significant since, for many of the social search related applications, predicting small

[2] To be precise, the bit consumption for roadNet in the first two columns is 18 bits per vertex, in the second 36, and in the third 54 bits per vertex.

distances accurately is of higher importance.

E. Landmark Bound Approach

Before comparing the performance of the landmark-based distance estimation functions that we proposed, we first studied the effect of the threshold parameter t on the accuracy of these learned functions. As described above, t is number of the best upper bounds that we use as features in in the distance function. In particular, we evaluated the effect of different values of t on the performance of our solutions, for landmark-sets L with sizes ranging from $|L| = 1$ to $|L| = 500$. Note that if $t < |L|$, then t is rounded down to $|L|$. As a result of these experiments, we verified our conjecture that considering more than just the tightest bound could lead to a more accurate distance estimation function. In addition to this, we observed that the benefit due to increasing the value of t becomes insignificant beyond $t = 100$. The plot of Figure 3 illustrates this effect for the Orkut graph and the f_{ub} function, and it is representative of the effect that we noticed for f_{ub} on other graphs as well; the benefit due to increasing t is less pronounced for f_{lb} (see [36]). The X-axis denotes the landmark set size $|L|$ and the Y-axis the average relative error for the particular landmark set size.

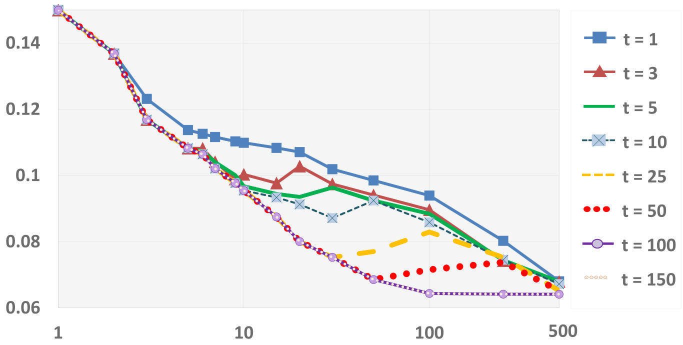

Fig. 3: Comparing various threshold values t. The X-axis denotes the landmark count and the Y-axis denotes the average relative error.

It is worth pointing out that our decision to use values of t greater than 1 was also strongly supported by our experiments, as the weights that were learned for the less tight bounds were often larger than those of the tightest bound. Also, the p-values for the coefficients of the less tight bounds were less than 10^{-4}, indicating that bounds other than the tightest one provided helpful signals.

For each landmark set size $|L|$, we trained a set of distance estimation functions. For each vertex pair in the training set we obtained the upper and lower bounds computed by L. Then we used the t tightest bounds (if $|L| < 100, t = |L|$, else $t = 100$) in order to create our feature vectors. Finally, we ran gradient descent and derived the $t + 1$ weights (one weight for each bound, and the intercept) of our distance function. The learning parameter that we used was $\eta = 0.005$, and the algorithm converged after approximately one thousand iterations.

Figure 4 provides a comparison between the performance of Base and the variations of our proposed distance estimation functions. The X-axis denotes the number of bits used by each node in order to store the $|L|$ distances to the landmarks, and the Y-axis corresponds to the average relative error. We studied the error of the distance estimation functions for various space constraints in order to understand their performance in different settings. In each plot of Figure 4, the line with the triangle markers represents the baseline method. The plain line represents f_{ub}, whereas the line with the circle markers is f_{lb}, and the dashed line is $f_{ub,lb}$. Finally, the dotted line is $f_{ub,lb,v}$, and it starts only after a few bits in the X-axis because we have introduced a constant space overhead for the representation of every vertex (since the values of its centrality metrics need to be stored).

For the social graphs, we observe that f_{ub} outperformed f_{lb}, while f_{lb} outperformed f_{ub} in the road network. As we mentioned earlier, our experiments on the known landmark-based distance-bounding methods revealed that their upper bounds were more accurate when it came to social graphs, and their lower bounds were more accurate for the road network. As a result it is to be expected that f_{lb} is more accurate for the road network than f_{ub}, as opposed to the social networks. Also, $f_{ub,lb}$ slightly outperforms both f_{ub} for the social graphs and f_{lb} for the road network, which confirms our intuition that combining upper with lower bounds could lead to better estimates than simply using one of the two. As expected for social networks, the coefficients assigned to lower bounds are much smaller than those assigned to upper bounds, and vice versa for the road network. The benefit of leveraging both bounds is actually clearer in the road network; this is due to the fact that lower bounds in social networks are not as informative signals about the actual distance as are upper bounds for the road network. Finally, we observe that while $f_{ub,lb,v}$ performs very poorly on the road network, its performance on all the social graphs is even better than that of $f_{ub,lb}$.

In addition to this, we observe that for most of the graphs our prediction methods significantly outperform Base; this is especially true when the space limitations are very strict. The Youtube graph is an exception, as the benefit from using the prediction method for the distance estimation is less significant. For all social graphs we can see that the error of the new distance estimation function is close to 0.1 with just a few landmarks (about 5 to 15 bits per node), whereas Base for the social graphs (except for Youtube) requires more than 200 bits per vertex to achieve the same accuracy. This means that $f_{ub,lb,v}$ requires almost 20 times less space to achieve the same accuracy as Base. Even for the roadNet-CA graph, $f_{ub,lb}$ requires only about 75 bits per vertex to bring the relative error down to 0.1, whereas Base requires more than 10 times more space to achieve the same accuracy. Moreover, just as we mentioned for the case of f_v, our experiments also indicate that the learned functions f_{ub}, $f_{ub,lb}$, and $f_{ub,lb,v}$ all perform better than the existing bounding-methods, especially for smaller distances [36], which is the case we are focusing on.

F. Query Processing Time

Computing the distance estimate for a pair of vertices (u, v) under this approach is very efficient and depends on the number of features that the distance estimation function uses. For f_v, query processing consists of retrieving the vertex features from an array in memory (even for graphs of $100M$ vertices

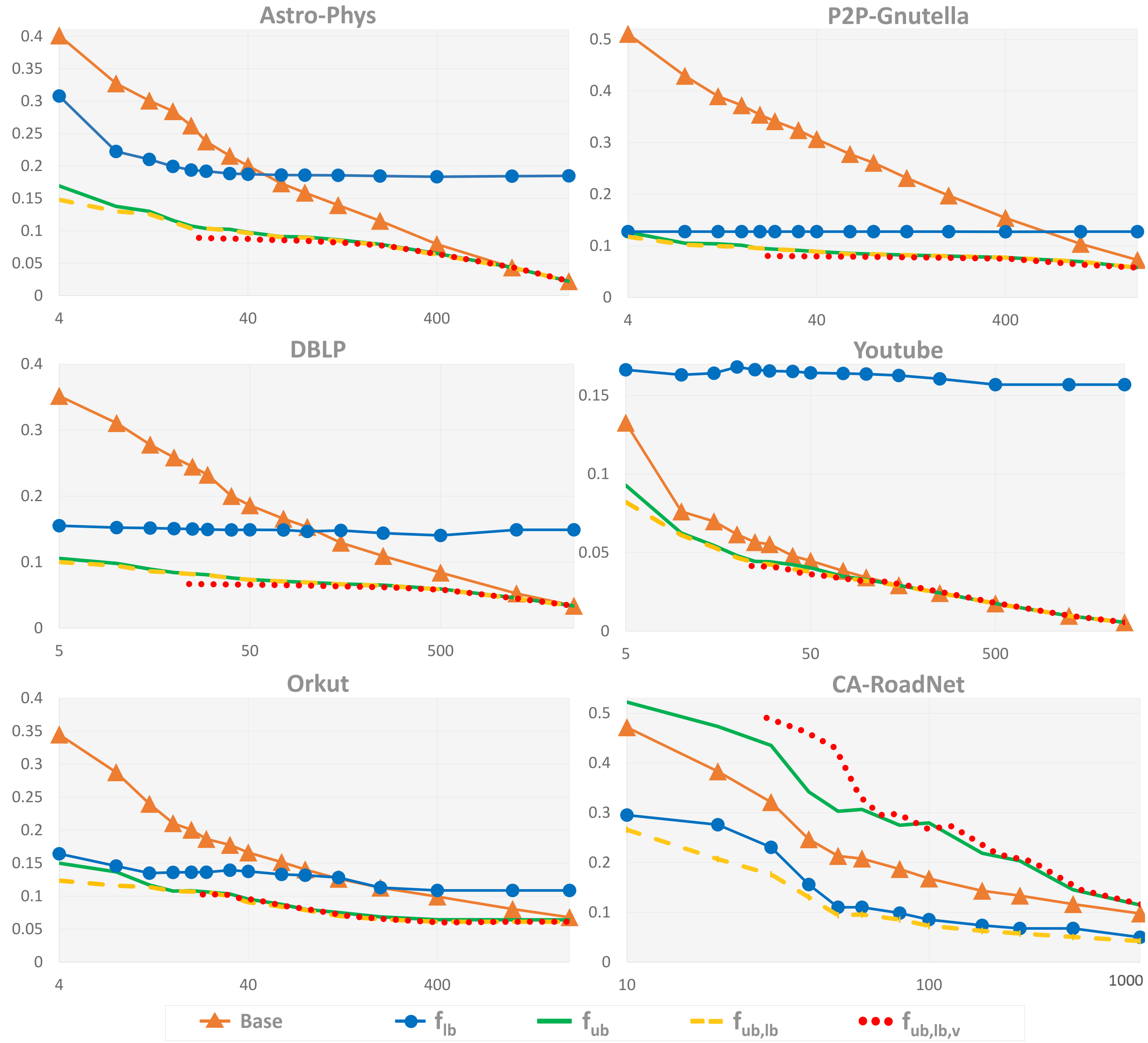

Fig. 4: Base vs. Distance Prediction Methods. The x-axis denotes bits per node, while the y-axis denotes the average relative error.

this structure fits in less than $2GB$) and then combining them using the learned weights. The performance depends on the size of the graph, since a series of random lookups in smaller arrays is faster than a series of lookups in larger arrays due to caching.

Graph Name	f_v	Base
Astro-Phys	0.04	0.312
DBLP	0.14	0.671
P2P-Gnutella	0.08	1.058
Orkut	0.22	0.48
Youtube	0.19	0.23
CA-RoadNet	0.2	0.17

TABLE III: Time (in μs) for f_v vs. Base for the same error.

Table III shows the query processing time for f_v and the query processing time of Base for the same average relative error that f_v achieves. Moreover, note that in [28] the reported time for Youtube was slightly larger than that of Base (for 20 landmarks) and had an error of 0.049. Our method has error 0.044 and needs 80% of the time that Base needs. This means that is has a smaller error, and requires in query processing time to achieve that accuracy.

For the case of f_{ub}, we require the same time as Base for the same number of landmarks to compute the bounds, but additional overhead is introduced to sort the bounds and compute their linear combination. However, due to the fact that f_{ub} achieves better accuracy with much fewer landmarks, given a limit on space the query processing the query processing time of f_{ub} is still smaller than that of Base.

VI. Conclusions

In this paper, we have studied the problem of estimating point-to-point distances in large graphs. We propose new solutions that significantly improve upon the accuracy of the state-of-the-art methods while maintaining very low space consumption and response time. From a different perspective, our methods achieve the same accuracy as landmark-based distance bounding methods with much faster query processing time and lower space requirements.

Our first solution uses pre-processing time in order to compute structural attributes of the graph nodes, and then leverages this information using linear regression in order to infer good distance predictions. We also propose a set of solutions that use the existing state-of-the-art methods as subroutines. Instead of estimating distances as bounds using landmarks, our machine-learned linear functions predict distances using these computed upper and lower bounds as features. Finally, we also consider combining the vertex attribute-based signals with the landmark-based bounds to get even better point-to-point distance estimates.

Acknowledgement

This work was supported by NSF Grant IIS-1117829 "Efficient Query Processing in Large Search Engines".

References

[1] A. Ukkonen, C. Castillo, D. Donato, and A. Gionis, "Searching the wikipedia with contextual information," *CIKM*, pp. 1351–1352, 2008.

[2] M. Potamias, F. Bonchi, C. Castillo, and A. Gionis, "Fast shortest path distance estimation in large networks," *CIKM*, pp. 867–876, 2009.

[3] B. Bahmani and A. Goel, "Partitioned multi-indexing: bringing order to social search," *WWW*, pp. 399–408, 2012.

[4] M. Qiao, L. Qin, H. Cheng, J. X. Yu, and W. Tian, "Top-k nearest keyword search on large graphs," *PVLDB*, vol. 6, no. 10, 2013.

[5] M. V. Vieira, B. M. Fonseca, R. Damazio, P. B. Golgher, D. de Castro Reis, and B. A. Ribeiro-Neto, "Efficient search ranking in social networks," *CIKM*, pp. 563–572, 2007.

[6] M. Qiao, H. Cheng, and J. X. Yu, "Querying shortest path distance with bounded errors in large graphs," *SSDBM*, vol. 6809, pp. 255–273, 2011.

[7] A. Gubichev, S. J. Bedathur, S. Seufert, and G. Weikum, "Fast and accurate estimation of shortest paths in large graphs," *CIKM*, pp. 499–508, 2010.

[8] A. D. Sarma, S. Gollapudi, M. Najork, and R. Panigrahy, "A sketch-based distance oracle for web-scale graphs," *WSDM*, pp. 401–410, 2010.

[9] E. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 1, no. 1, pp. 269–271, 1959.

[10] A. V. Goldberg, H. Kaplan, and R. F. F. Werneck, "Reach for a*: Effcient point-to-point shortest path algorithms," Microsoft Research, CA, Tech. Rep. MSR-TR-2005-132, October 2005.

[11] A. V. Goldberg and C. Harrelson, "Computing the shortest path: $A*$ search meets graph theory," *SODA*, pp. 156–165, 2005.

[12] A. V. Goldberg, "Point-to-point shortest path algorithms with preprocessing," *SOFSEM (1)*, vol. 4362, pp. 88–102, 2007.

[13] C. Demetrescu, A. V. Goldberg, and D. S. Johnson, "Implementation challenge for shortest paths," in *Encyclopedia of Algorithms*, M.-Y. Kao, Ed. Springer, 2008.

[14] Y. Xiao, W. Wu, J. Pei, W. Wang, and Z. He, "Efficiently indexing shortest paths by exploiting symmetry in graphs," *EDBT*, pp. 493–504, 2009.

[15] E. Cohen, E. Halperin, H. Kaplan, and U. Zwick, "Reachability and distance queries via 2-hop labels," *SIAM J. Comput.*, vol. 32, no. 5, pp. 1338–1355, 2003.

[16] R. Jin, N. Ruan, Y. Xiang, and V. Lee, "A highway-centric labeling approach for answering distance queries on large sparse graphs," in *SIGMOD*, 2012, pp. 445–456.

[17] A. W.-C. Fu, H. Wu, J. Cheng, and R. C.-W. Wong, "Is-label: An independent-set based labeling scheme for point-to-point distance querying," *Proc. VLDB Endow.*, vol. 6, no. 6, pp. 457–468, 2013.

[18] T. Akiba, Y. Iwata, and Y. Yoshida, "Fast exact shortest-path distance queries on large networks by pruned landmark labeling," in *SIGMOD*, 2013, pp. 349–360.

[19] M. Thorup and U. Zwick, "Approximate distance oracles," *STOC*, pp. 183–192, 2001.

[20] J. Bourgain, "On lipschitz embedding of finite metric spaces in hilbert space," *Israel Journal of Mathematics*, vol. 52, no. 1, pp. 46–52, 1985.

[21] J. Matousek, "On the distortion required for embedding finite metric spaces into normed spaces," *Israel Journal of Mathematics*, vol. 93, no. 1, pp. 333–344, 1996.

[22] Y. Bartal, "On approximating arbitrary metrices by tree metrics," *STOC*, pp. 161–168, 1998.

[23] J. Fakcharoenphol, S. Rao, and K. Talwar, "A tight bound on approximating arbitrary metrics by tree metrics," *J. Comput. Syst. Sci.*, vol. 69, no. 3, pp. 485–497, 2004.

[24] T.-H. H. Chan, K. Dhamdhere, A. Gupta, J. M. Kleinberg, and A. Slivkins, "Metric embeddings with relaxed guarantees," *SIAM J. Comput.*, vol. 38, no. 6, pp. 2303–2329, 2009.

[25] X. Zhao, A. Sala, H. Zheng, and B. Y. Zhao, "Efficient shortest paths on massive social graphs," *CollaborateCom*, pp. 77–86, 2011.

[26] S. Baswana, "Streaming algorithm for graph spanners - single pass and constant processing time per edge," *Inf. Process. Lett.*, vol. 106, no. 3, pp. 110–114, 2008.

[27] J. Feigenbaum, S. Kannan, A. McGregor, S. Suri, and J. Zhang, "Graph distances in the streaming model: the value of space," *SODA*, pp. 745–754, 2005.

[28] M. Qiao, H. Cheng, L. Chang, and J. Yu, "Approximate shortest distance computing: A query-dependent local landmark scheme," *IEEE Transactions on Knowledge and Data Engineering*, vol. 26, no. 1, pp. 55–68, 2014.

[29] K. Tretyakov, A. Armas-Cervantes, L. García-Bañuelos, J. Vilo, and M. Dumas, "Fast fully dynamic landmark-based estimation of shortest path distances in very large graphs," *CIKM*, pp. 1785–1794, 2011.

[30] R. Jin, Y. Xiang, N. Ruan, and D. Fuhry, "3-hop: a high-compression indexing scheme for reachability query," *SIGMOD Conference*, pp. 813–826, 2009.

[31] R. Jin, Y. Xiang, N. Ruan, and H. Wang, "Efficiently answering reachability queries on very large directed graphs," *SIGMOD Conference*, pp. 595–608, 2008.

[32] S. Trißl and U. Leser, "Fast and practical indexing and querying of very large graphs," *SIGMOD Conference*, pp. 845–856, 2007.

[33] R. Schenkel, A. Theobald, and G. Weikum, "Hopi: An efficient connection index for complex xml document collections," *EDBT*, vol. 2992, pp. 237–255, 2004.

[34] B. Shaw, B. C. Huang, and T. Jebara, "Learning a distance metric from a network," *NIPS*, pp. 1899–1907, 2011.

[35] U. Brandes, "A faster algorithm for betweeness centrality," *Journal of Mathematical Sociology*, vol. 25, no. 2, pp. 163–177, 2001.

[36] M. Christoforaki and T. Suel, "Learning to estimate pairwise distances in large graphs," http://cse.poly.edu/~christom/distanceestimation.pdf 2014.