Fast Shortest Path Distance Estimation in Large Networks

Michalis Potamias[1*] Francesco Bonchi[2] Carlos Castillo[2] Aristides Gionis[2]

[1]Computer Science Department
Boston University, USA
mp@cs.bu.edu

[2]Yahoo! Research
Barcelona, Spain
{bonchi,chato,gionis}@yahoo-inc.com

ABSTRACT

In this paper we study approximate *landmark-based* methods for point-to-point distance estimation in very large networks. These methods involve selecting a subset of nodes as landmarks and computing offline the distances from each node in the graph to those landmarks. At runtime, when the distance between a pair of nodes is needed, it can be estimated quickly by combining the precomputed distances.

We prove that selecting the *optimal* set of landmarks is an **NP**-hard problem, and thus heuristic solutions need to be employed. We therefore explore theoretical insights to devise a variety of simple methods that scale well in very large networks. The efficiency of the suggested techniques is tested experimentally using five real-world graphs having millions of edges.

While theoretical bounds support the claim that random landmarks work well in practice, our extensive experimentation shows that smart landmark selection can yield dramatically more accurate results: for a given target accuracy, our methods require as much as 250 times less space than selecting landmarks at random. In addition, we demonstrate that at a very small accuracy loss our techniques are several orders of magnitude faster than the state-of-the-art exact methods. Finally, we study an application of our methods to the task of social search in large graphs.

Categories and Subject Descriptors H.4.3 [**Information Systems Applications**]: Communications Applications
General Terms Algorithms
Keywords Graphs, shortest-paths, landmarks methods.

1. INTRODUCTION

Understanding the mechanisms underlying the characteristics and the evolution of complex networks is an important task, which has received interest by various disciplines including sociology, biology, and physics. In the last years we have witnessed a continuously increasing availability of very large networks; blogs, sites with user-generated content, social networks, and instant messaging systems nowadays count hundreds of millions of users that are active on a daily basis. For graphs of this size, even seemingly simple algorithmic problems become challenging tasks.

One basic operation in networks is to measure how close one node is to another, and one intuitive network distance is the *geodesic distance* or *shortest-path distance*. Computing shortest-path distances among nodes in a graph is an important primitive in a variety of applications including, among many others, social-network analysis, VLSI design in electronics, protein interaction networks in biology and route computation in transportation.

Recently, new motivating applications have arisen in the context of web search and social networks. In web search, the distance of a query's initiation point (the query context) to the relevant web-pages could be an important aspect in the ranking of the results [34]. In social networks, a user may be interested in finding other users, or in finding content from users that are close to her in the social graph [30]. This *socially sensitive* search model has been suggested as part of the social network search experience [2, 36]. Using the shortest path distance as a primitive in ranking functions for search tasks is the main motivation of our work.

Although computing shortest paths is a well studied problem, exact algorithms can not be adopted for nowadays real-world massive networks, especially in online applications where the distance must be provided in the order of a few milliseconds. A full breadth-first search (BFS) traversal of a web-graph of 4 M nodes and 50 M edges takes roughly a minute in a standard modern desktop computer.[1] For the same graph the best known point-to-point shortest path algorithms that combine Dijkstra with A* and landmarks, require to access an average of 20 K nodes in order to determine the shortest path between two nodes. On the other hand, precomputing all the shortest paths and storing them explicitly is infeasible: one would need to store a matrix of approximately 12 trillion elements.

The methods described in this paper use precomputed information to provide fast estimates of the actual distance in very short time. The offline step consists of choosing a subset of nodes as *landmarks* (or *reference objects*) and computing distances from every node to them. Such precomputed distance information is often referred to as an *embedding*.

Our contribution. In this paper we present an extensive analysis of various strategies for selecting landmarks. We

*Part of this work was done while the first author was visiting Yahoo! Research Barcelona, under the Yahoo! internship program.

[1]Using the BFS routine of C++ Boost library.

CIKM'09, November 2–6, 2009, Hong Kong, China.

devise and experimentally compare more than 30 strategies that scale well to very large networks. For presentation sake, we report the best ones; in case of ties we report the simplest. Our experimentation shows that for a given target accuracy, our techniques require orders of magnitude less space than random landmark selection, allowing an efficient approximate computation of shortest path distances among nodes in very large graphs. To the best of our knowledge, this is the first systematic analysis of scalable landmarks selection strategies for shortest-path computation in large networks. Our main contributions are summarized as follows:

- We define the problem of optimal landmark selection in a graph and prove that it is **NP**-hard (Section 3).
- We theoretically motivate and suggest simple and intuitive strategies to choose landmarks that scale well to huge graphs (Section 4).
- We demonstrate the effectiveness and robustness of our techniques experimentally using five large different real-world networks with millions of edges (Section 5).
- We report significant efficiency gains with respect to the state-of-the-art. We study the *triangulation* performance of landmark selection methods (Section 5.4). Although theoretical bounds have been proven in literature [24] for random landmarks selection we prove empirically that our methods outperform it by large margins on real networks. With respect to exact algorithms we show that several orders of magnitude in efficiency can be traded-off with a very small loss in accuracy using the proposed techniques (Section 5.5).
- We apply our methods to *social search* (Section 6), showing high precision and accuracy in the problem of finding the closest nodes in the graph that match a given query.

In our experimental evaluation we use real world networks: social graphs with explicit or implicit links from Flickr, a graph based on the communication network of the Yahoo! Instant Messenger service, and the coauthorship graph from the DBLP records. We also use a web-graph defined by the Wikipedia pages and their hyperlinks.

Paper structure. In Section 2 we outline the related work, and in Section 3 we introduce our notation and our algorithmic framework. Section 4 presents a series of landmark section strategies which are experimentally evaluated in Section 5. Section 6 evaluates an application of our methods for fast social-network-aware search. Finally, Section 7 presents some concluding remarks.

2. RELATED WORK

Exact shortest-path distances. Dijkstra described the algorithm to compute single source shortest paths (SSSP) in weighted graphs with n nodes and m edges from a node to all others [11]. The cost is $O(n^2)$ in general and can be reduced to $O(m+n\log n)$ for sparse graphs. For unweighted graphs, shortest paths can be computed using Breadth First Search (BFS) in time $O(m+n)$. Floyd-Warshall algorithm employs dynamic programming to solve the all-pairs shortest paths (APSP) problem in an elegant and intuitive way [13] in time $O(n^3)$. Still, the complexity of computing APSP by invoking n Dijkstra/BFS computations is asymptotically faster, since it costs $O(nm+n^2\log n)$ and $O(nm)$ respectively.

The state of the art in point to point shortest path (PPSP) queries involves combining bidirectional Dijkstra with A* and lower bounds (ALT algorithms) [16, 17, 37, 27, 22]. ALT algorithms employ landmarks in order to prune the search space of the shortest path computation. Their landmarks are similar to the ones we experimented with for the lower-bound estimation (see Section 4.4); instead, in this paper, we use heuristics for selecting landmarks that work well with upper-bound estimates. Our paper addresses a different problem than the one in [16, 17, 37, 27, 22] since we are interested only on the length of a shortest path, not the path itself. Thus, we avoid any kind of online Dijkstra/BFS traversals of the graph. However, to demonstrate the savings that one can obtain if the path itself is not of interest, and if only the distance length between two nodes is important, in Section 5 we compare our method to these state-of-the-art techniques and demonstrate that orders of magnitude in efficiency can be gained with a very small loss in accuracy.

Indexing for approximate shortest-paths. We are interested in preprocessing a graph so that PPSP queries can be answered approximately and quickly at runtime. Thorup and Zwick [32] observe that this problem is probably the most natural formulation of the APSP. In their paper they obtain the result that for any integer $k \geq 1$ a graph can be preprocessed in $O(kmn^{\frac{1}{k}})$ expected time, using a data structure of size $O(kn^{1+\frac{1}{k}})$, and a PPSP query can be processed in time $O(k)$. The quotient of the division of the estimated distance and the exact is guaranteed to lie within $[1, 2k-1]$. For $k=1$ we get the exact solution of computing all shortest paths and storing them, which is prohibitively expensive.

For $k=2$ the estimate may be three times larger than the actual distance. In large real-world graphs this bound is already problematic because distances are short due to the *small-world phenomenon*. In a small-world network, such as the Flickr-contacts graph described in Section 5, for an estimated distance of 6, the exact distance is only guaranteed to lie within the interval $[2, 6]$, along with almost every pairwise distance in this graph. A survey on exact and approximate distances in graphs can be found in [38].

Embedding methods. Our work is related to general embedding methods. In domains with a computationally expensive distance function, significant speed-ups can be obtained by embedding objects into another space and using a more efficient distance function, such as an L_p norm. Several methods have been proposed to embed a space into a Euclidean space [6, 20]. There have been attempts to optimize the selection of reference objects for such a setting [3, 35]. Other dimensionality reduction techniques are also widely studied especially in theory and machine learning.

Landmarks have already been used for graph measurements in many applications [10, 26, 31, 28] such as round-trip propagation and transmission delay in networks: however, how to optimally select the location of landmarks has not been extensively studied.

Kleinberg et al. [24] discuss the problem of approximating network distances in real networks via embeddings using a small set of *beacons* (i.e., landmarks). Of most interest is the fact that they introduce in their analysis the notion of *slack*, as a fraction of pairs in the network for which the

algorithm provides no guarantees. Their analysis considers choosing beacons randomly. In this paper we show that in practice, simple intuitive strategies work much better than the random. Abraham et al. [1] generalize the metric embedding with slack. On another perspective, computing shortest paths in spatial networks has also attracted interest recently [25, 29]; our work is different since we focus on graphs that exhibit complex social network or web-graph behavior.

Applications. Our work is also tightly connected to the various notions that have been introduced to measure the *centrality* of a vertex. Betweenness centrality measures the amount of shortest paths passing from a vertex while closeness centrality measures the average distance of a vertex to all other vertices in the network [15]. Brandes [7] gave the best known algorithm to compute the exact betweenness centrality of all vertices by adapting the APSP Dijkstra algorithm. The algorithm runs in $O(nm+n^2 logn)$ time, which is prohibitive for large graphs. Bader et al. [4] gave a sampling-based approximation algorithm and showed that centrality is easier to approximate for central nodes. In our work, we use closeness centrality as a strategy in choosing central points as landmarks in the graph.

Fast PPSP computation is becoming very relevant for Information Retrieval. Socially sensitive search in social networks and location-aware search are attracting a substantial interest in the information retrieval community [5]. For instance, it has been found that people who chat with each other are more likely to share interests [30]. An experiment discussed in Section 5 considers ranking search results in social networks based on shortest path distances. This problem has also been studied recently by Vieira et al. [36]. Their work is also based on landmarks, but their landmarks are chosen randomly. Since our work has been inspired by the task of *social search* we revisit it in Section 6 and show that our techniques outperform the random landmark selection [36] by very large margins.

Approximation methods for computing other graph proximity functions that are based on random walks, such as personalized pagerank and random walk with restart, have also been studied recently [14, 33]. The growing interest in involving context and/or social connections in search tasks, suggests that distance computation will soon be a primitive of ranking functions. The restriction is that the ranking functions of search engines have hard computational deadlines to meet, in the order of hundreds of milliseconds. Our methods can provide accurate results within these deadlines.

3. ALGORITHMIC FRAMEWORK

In this section we introduce the notation that we use in the rest of the paper. We then describe how to index distances very efficiently using landmarks. We formally define the landmark-selection problem that we consider in this paper, and prove that it is an **NP**-hard problem.

3.1 Notation

Consider a graph $G(V, E)$ with n vertices and m edges. Given two vertices $s, t \in V$, define $\pi_{s,t} = \langle s, u_1, u_2, ..., u_{\ell-1}, t\rangle$ to be a path of length $|\pi_{s,t}| = \ell$ between s and t, if $\{u_1, ..., u_\ell\} \subseteq V$ and $\{(s, u_1), (u_1, u_2), \ldots, (u_{\ell-1}, t)\} \subseteq E$, and let $\Pi_{s,t}$ be the set of all paths from s to t. Accordingly, let $d_G(s, t)$ be the length of the shortest path between any two vertices $s, t \in V$, we refer to this as the geodesic distance, or distance, between such vertices. In other words, $d_G(s, t) = |\pi^*_{s,t}| \leq |\pi_{s,t}|$ for all paths $\pi_{s,t} \in \Pi_{s,t}$. Let $\mathrm{SP}_{s,t}$ be the set of paths whose length is equal to $d_G(s, t)$.

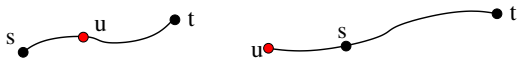

Figure 1: Illustration of the cases for obtaining tight upper bounds (left) and tight lower bounds (right) as provided by Observations 1 and 2

For simplicity we consider unweighted, undirected graphs, but all the ideas in our paper can be easily applied to weighted and/or directed graphs.

Consider an ordered set of d vertices $D = \langle u_1, u_2, \ldots, u_d\rangle$ of the graph G, which we call *landmarks*. The main idea is to represent each other vertex in the graph as a vector of shortest path distances to the set of landmarks. This is also called an *embedding* of the graph. In particular, each vertex $v \in V$ is represented as a d-dimensional vector $\phi(v)$:

$$\phi(v) = \langle d_G(v, u_1), d_G(v, u_2), ..., d_G(v, u_d)\rangle \quad (1)$$

For ease of presentation, from now on we will denote the i-th coordinate of $\phi(v)$ by v_i, i.e., $v_i = d_G(v, u_i)$.

3.2 Distance bounds

The shortest-path distance in graphs is a metric, and therefore it satisfies the triangle inequality. That is, given any three nodes s, u, and t, the following inequalities hold.

$$d_G(s, t) \leq d_G(s, u) + d_G(u, t), \quad (2)$$

$$d_G(s, t) \geq |d_G(s, u) - d_G(u, t)| \quad (3)$$

An important observation that we will use to formulate the landmark-selection problem is that if u belongs to one of the shortest paths from s to t, then the inequality (2) holds with equality.

Observation 1. *Let s, t, u be vertices of G. If there exists a path $\pi_{s,t} \in \mathrm{SP}_{s,t}$ so that $u \in \pi_{s,t}$ then $d_G(s, t) = d_G(s, u) + d_G(u, t)$.*

A similar condition exists for the inequality (3) to be tight, but in this case, it is required that either s or t are the "middle" nodes.

Observation 2. *Let s, t, u be vertices of G. If there exists a path $\pi_{s,u} \in \mathrm{SP}_{s,u}$ so that $t \in \pi_{s,u}$, or there exists a path $\pi_{t,u} \in \mathrm{SP}_{t,u}$ so that $s \in \pi_{t,u}$, then $d_G(s, t) = |d_G(s, u) - d_G(u, t)|$.*

The situation described in Observations 1 and 2 is shown in Figure 1.

3.3 Using landmarks

Given a graph G with n vertices and m edges, and a set of d landmarks D, we precompute the distances between each vertex in G and each landmark. The cost of this offline computation is d BFS traversals of the graph: $O(md)$.

Recall that our task is to compute $d_G(s, t)$ for any two vertices $s, t \in V$. Due to Inequalities (2) and (3), we have

$$\max_i |s_i - t_i| \leq d_G(s, t) \leq \min_j \{s_j + t_j\}.$$

In other words, the true distance $d_G(s, t)$ lies in the range $[L, U]$, where $L = \max_i |t_i - s_i|$ and $U = \min_j \{s_j + t_j\}$.

Notice that one landmark may provide the best lower bound and another the best upper bound. Any value in the range $[L, U]$ can be used as an estimate $\tilde{d}(s,t)$ for the real value of $d_G(s,t)$. Some choices include using the upper bound

$$\tilde{d}_u(s,t) = U,$$

using the lower bound

$$\tilde{d}_l(s,t) = L,$$

the middle point

$$\tilde{d}_m(s,t) = \frac{L+U}{2},$$

or the geometric mean

$$\tilde{d}_g(s,t) = \sqrt{L \cdot U}.$$

Notice that in all cases the estimation is very fast, as only $O(d)$ operations need to be performed, and d can be thought of as being a constant, or a logarithmic function of the size of the graph.

Our experiments indicate that the "upper bound" estimates $\tilde{d}_u(s,t) = U$ work much better than the other types of estimates, so, in the rest of the paper, we focus on the upper-bound estimates. We only comment briefly on lower-bound estimates later in Section 4.4.

As follows by Observation 1, the approximation $\tilde{d}_u(s,t)$ is exact if there exists a landmark in D, which is also in a shortest path from s to t. This motivates the definition of *coverage*:

DEFINITION 1. *We say that a set of landmarks D* covers *a specific pair of vertices (s,t) if there exists at least one landmark in D that lies in one shortest path from s to t.*

Our landmark-selection problem is formulated as follows.

PROBLEM 1 (LANDMARKS$_d$). *Given a graph $G = (V, E)$ select a set of d landmarks $D \subseteq V$ so that the number of pairs of vertices $(s,t) \in V \times V$ covered by D is maximized.*

A related problem is the following

PROBLEM 2 (LANDMARKS-COVER). *Given a graph $G = (V, E)$ select the minimum number of landmarks $D \subseteq V$ so that all pairs of vertices $(s,t) \in V \times V$ are covered.*

3.4 Selecting good landmarks

To obtain some intuition about landmark selection, consider the LANDMARKS$_d$ problem for $d = 1$. The best landmark to select, is a vertex that it is very *central* in the graph, and many shortest paths pass through it. In fact, selecting the best landmark is related to finding the vertex with the highest *betweenness centrality* [15].

To remind the reader the definition of betweenness centrality, given two vertices s and t, let σ_{st} denote the number of shortest paths from s to t. Also let $\sigma_{st}(u)$ denote the number of shortest paths from s to t that some $u \in V$ lies on. The betweenness centrality of the vertex u is defined as

$$C_B(u) = \sum_{s \neq u \neq t \in V} \frac{\sigma_{st}(u)}{\sigma_{st}} \quad (4)$$

The fastest known algorithms to compute betweenness centrality exactly are described by Brandes [7]. They extend well-known all-pairs-shortest-paths algorithms [9]. The time cost $O(nm)$ for unweighted graphs and $O(nm + n^2 \log n)$ for weighted graphs. Additionally, Bader et al. [4] discuss how to approximate betweenness centrality by random sampling.

For our problem, consider a modified definition of betweenness centrality according to which we define $I_{st}(u)$ to be 1 if u lies on at least one shortest path from s to t, and 0 otherwise. We then define

$$C(u) = \sum_{s \neq u \neq t \in V} I_{st}(u). \quad (5)$$

It follows immediately that the optimal landmark for the LANDMARKS$_d$ problem with $d = 1$ is the vertex that maximizes $C(u)$. Our modified version $C(u)$ can be computed as efficiently as $C_B(u)$ by modifying Brandes' algorithm [7].

3.5 Problem complexity and approximation algorithms

Both of the problems LANDMARKS$_d$ and LANDMARKS-COVER are **NP**-hard. An easy reduction for the LANDMARKS-COVER problem can be obtained from the VERTEX-COVER problem.

THEOREM 1. LANDMARKS-COVER *is* **NP**-*hard.*

PROOF. We consider the decision version of the problems LANDMARKS-COVER and VERTEX-COVER. The latter problem is defined as follows: given a graph G, and an integer k, decide if there is a subset of vertices $V' \subseteq V$ of size at most k so that for all edges $(u,v) \in E$ either $u \in V'$ or $v \in V'$. Transform an instance of VERTEX-COVER to an instance of LANDMARKS-COVER. Consider a solution D for LANDMARKS-COVER. Consider now the set of all 1-hop neighbors and observe that each pair is connected by a unique shortest path of length 1 (i.e. an edge). Since all pairs of vertices are covered, so are 1-hop neighbors, therefore the edges of E are also covered by D, therefore, D is a solution to VERTEX-COVER. Conversely, consider a solution V' for VERTEX-COVER. Consider a pair of vertices $(s,t) \in V \times V$, and any shortest path $\pi_{s,t}$ between them. Some vertices of V' should be on the edges of the path $\pi_{s,t}$, and therefore V' is also a solution to LANDMARKS-COVER. □

As a consequence, LANDMARKS$_d$ is also **NP**-hard.

Next we describe a polynomial-time approximation solution to the landmark-selection problem. The main idea is to map the problem to SET-COVER problem. Given the graph $G = (V, E)$, we consider a set of elements $U = V \times V$ and a collection of sets $\mathcal{S}$, so that each set $S_v \in \mathcal{S}$ corresponds to a vertex $v \in V$. A set S_v contains an element $(s,t) \in U$ if v lies on a shortest path from s to v. Thus, solving the SET-COVER problem on $(U, \mathcal{S})$ with the greedy algorithm [8], we obtain a $O(\log n)$-approximation to LANDMARKS-COVER problem and a $(1 - 1/e)$-approximation to LANDMARKS$_d$ problem.

However, the running time of the above approximation algorithm is $O(n^3)$, which is unacceptable for the size of graphs that we consider in this paper.

The suggested strategies of the next section are motivated by the observations made in this section regarding properties of good landmarks.

4. LANDMARK-SELECTION STRATEGIES

This section describes our landmark-selection strategies.

The baseline scalable strategy is to select landmarks at random [36, 24, 31]. The strategies we propose are motivated by the discussion in the previous section. On a high

level, the idea is to select as landmarks "central" nodes of the graph, so that many shortest paths are passing through. We use two proxies for selecting central nodes: (*i*) high-degree nodes and (*ii*) nodes with low *closeness centrality*, where the closeness centrality of a node u is defined as the average distance $\frac{1}{n}\sum_v d_G(u,v)$ of u to other nodes in the graph.

In order to cover many different pairs of nodes, we need to *spread* the landmarks throughout the graph. In accordance, we propose two improvements for our strategies: (*i*) a *constrained* variant, where we do not select landmarks that are too close to each other, and (*ii*) a *partitioning* variant, where we first partition the graph and then select landmarks from different partitions.

4.1 Basic strategies

RANDOM**:** The baseline landmark-selection strategy consists of sampling a set of d nodes uniformly at random from the graph.

DEGREE**:** We sort the nodes of the graph by decreasing degree and we choose the top d nodes. Intuitively, the more connected a node is, the higher the chance that it participates in many shortest paths.

CENTRALITY**:** We select as landmarks the d nodes with the lowest closeness centrality. The intuition is that the closer a node appears to the rest of the nodes the bigger the chance that it is part of many shortest paths.

Computing the closeness centrality for all nodes in a graph is an expensive task, so in order to make this strategy scalable to very large graphs, we resort to computing centralities approximately. Our approximation works by selecting a sample of random seed nodes, performing a BFS computation from each of those seed nodes, and recording the distance of each node to the seed nodes. Since the seeds are selected uniformly at random and assuming that graph distances are bounded by a small number (which is true since real graphs typically have small diameter), we can use the Hoeffding inequality [21] to show that we can obtain arbitrarily good approximation to centrality by sampling a constant number of seeds.

4.2 Constrained strategies

Our goal is to cover as many pairs as possible. Using a basic strategy such as the ones described above, it may occur that the second landmark we choose covers a set of pairs that is similar to the one covered by the first, and thus its contribution to the cover is small.

The constrained variant of our strategies depends on a depth parameter h. We first rank the nodes according to some strategy (e.g., highest degree or lowest closeness centrality). We then select landmarks iteratively according to their rank. For each landmark l selected, we discard from consideration all nodes that are at distance h or less from l. The process is repeated until we select d landmarks.

We denote our modified strategies by DEGREE/h and CENTRALITY/h.

For the experiments reported in the next section we use $h = 1, 2, 3$ and we obtain the best results for $h = 1$. So, in the rest of the paper, we only consider this latter case.

4.3 Partitioning-based strategies

In order to spread the landmarks across different parts of the graph, we also suggest partitioning the graph using a fast graph-partitioning algorithm (such as METIS [23]) and then select landmarks from the different partitions. We suggest the following partitioning-based strategies.

DEGREE/P**:** Pick the node with the highest degree in each partition.

CENTRALITY/P**:** Pick the node with the lowest centrality in each partition.

BORDER/P**:** Pick nodes close to the *border* of each partition. We do so by picking the node u with the largest $b(u)$ in each partition, according to the following formula:

$$b(u) = \sum_{i \in P, u \in p, i \neq p} d_i(u) \cdot d_p(u), \tag{6}$$

where P is the set of all partitions, p is the partition that node u belongs to, and $d_i(u)$ is the degree of u with respect to partition i (i.e., the number of neighbors of u that lie in partition i). The intuition of the above formula is that if a term $d_i(u) \cdot d_p(u)$ is large, then node u lies potentially among many paths from nodes s in partition i to nodes t in partition p: such (s,t) pairs of nodes have distance at most 2, and since they belong to different clusters most likely there are not direct edges for most of them. For completeness, we remark that our experiments in all graphs, indicate that no significant improvement can be obtained by more complex strategies that combine both partitioning and constrained strategies.

4.4 Estimates using the lower bounds

As mentioned in Section 3.3, values $\tilde{d}_l(s,t)$, $\tilde{d}_m(s,t)$, and $\tilde{d}_g(s,t)$ can also be used for obtaining estimates for the shortest path length $d_G(s,t)$.

Following Observation 2, landmarks that give good lower-bound estimates $\tilde{d}_l(s,t)$ are nodes on the "periphery" of the graph, so that many graph nodes are on a shortest path between those landmarks and other nodes. Most of the strategies we discuss above are optimized to give good upper-bound landmarks, by selecting central nodes in the graph, and they perform poorly for lower-bound landmarks.

In fact, random landmarks perform better than any of the above methods with respect to lower bounds. With the intuition to select landmarks on the periphery of the graph, we also tried variations of the following algorithm (also described in [17]): (*i*) select the first landmark at random (*ii*) iteratively perform a BFS from the last selected landmark and select the next landmark that is the farthest away from all selected landmarks so far (e.g., maximizing the minimum distance to a selected landmark). This algorithm performs better than selecting landmarks at random, but overall the performance is still much worse than any of the methods for upper-bound landmarks.

5. EXPERIMENTAL EVALUATION

We present experimental results in terms of efficiency, accuracy and comparison to existing work for five datasets.

5.1 Datasets

In order to demonstrate the robustness of our methods and to show their performance in practice, we present experiments with five real-world datasets. The first four are anonymized datasets obtained from various sources, namely Flickr, Yahoo! Instant Messenger (Y!IM), and DBLP. The last one is a document graph from the Wikipedia (nodes

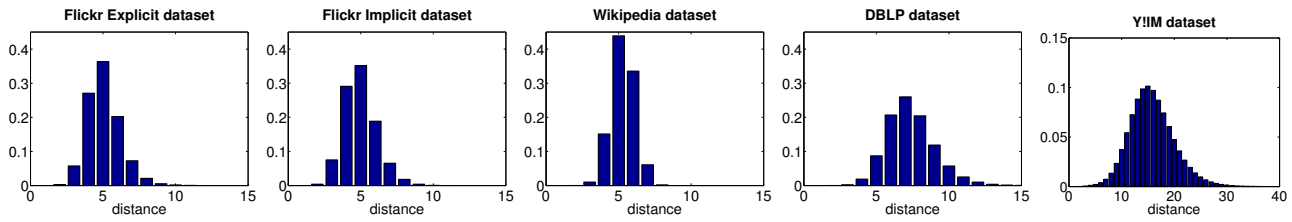

Figure 2: Distributions of distances in our datasets.

are articles, edges are hyperlinks among them). Figure 2 illustrates the distance distributions. Next, we provide more details and statistics about the datasets.

Flickr-E: Explicit contacts in Flickr. Flickr is a popular online-community for sharing photos, with millions of users. The first graph we construct is representative of its social network, in which the node set V represent users, an the edges set E is such that $(u, v) \in E$ if and only if a user u has added user v as his/her contact.

We use a sample of Flickr which with 25M users and 71M relationships. In order to create a sub-graph suitable for our experimentation we perform the following steps. First, we create a graph from Flickr by taking all the contact relationships that are reciprocal. Then, we keep all the users in the US, UK, and Canada. For all of our datasets, we take the largest connected component of the final graph.

Flickr-I: Implicit contacts in Flickr. This graph infers user relationships by observing user behavior. *Reciprocal comments* are used as a proxy for shared interest. In this graph an edge $(u, v) \in E$ exists if and only if a user u has commented on a photo of v, and v has commented on a photo by u.

DBLP coauthors graph. We extract the DBLP coauthors graph from a recent snapshot of the DBLP database that considers only the journal publications. There is an undirected edge between two authors if they have coauthored a journal paper.

Yahoo! IM graph. We use a subgraph of the Yahoo! Instant Messenger contact graph, containing only users who are active also in Yahoo! Movies. This makes this graph much sparser than the others. Goyal et al. describe this dataset in detail [18].

Wikipedia hyperlinks. Apart from the previous four datasets, which are social and coauthorship graphs, we consider an example of a web graph, the Wikipedia link graph. This graph represents Wikipedia pages that link to one another. We consider all hyperlinks as undirected edges. We remove pages having more than 500 hyperlinks, as they are mostly lists.

Summary statistics about these datasets are presented in Table 1. The statistics include the effective diameter $\ell_{0.9}$ and the median diameter $\ell_{0.5}$, which are the minimum shortest-path distances at which 90% and 50% of the nodes are found respectively, and the clustering coefficient c. In these graphs the degree follows a Zipf distribution in which the probability of having degree x is proportional to $x^{-\theta}$; the parameter θ fitted using Hill's estimator [19] is also shown in the table.

5.2 Approximation quality

We measure the accuracy of our methods in calculating shortest paths between pairs of nodes. We randomly choose 500 random pairs of nodes. In the case of the RANDOM selection strategy we average the results over 10 runs for each landmark set size. We report for each method and dataset the average of the approximation error: $|\hat{\ell} - \ell|/\ell$ where ℓ is the actual distance and $\hat{\ell}$ the approximation.

Figure 3 shows representative results for three datasets. Observe that using two landmarks chosen with the CENTRALITY strategy in the Flickr-E dataset yields an approximation equal to the one provided by using 500 landmarks selected by RANDOM. In terms of space and query-time this results in savings of a factor of 250. For the DBLP dataset the respective savings are of a factor greater than 25.

Table 2 summarizes the approximation error of the strategies across all 5 datasets studied here. We are using two landmark sizes: 20 and 100 landmarks; with 100 landmarks we see error rates of 10% or less across most datasets.

By examining Table 2 one can conclude that even simple strategies are much better than random landmark selection. Selecting landmarks by DEGREE is a good strategy, but sometimes does not perform well, as in the case of the Y!IM graph in which it can be worse than random. The strategies based on CENTRALITY yield good results across all datasets.

5.3 Computational efficiency

We implemented our methods in C++ using the Boost and STL Libraries.All the experiments are run on a Linux server with 8 1.86GHz Intel Xeon processors and 16GB of memory.

Regarding the online step the tradeoffs are remarkable: The online step is constant, $O(d)$ per pair where d is the number of landmarks. In practice it takes less than a millisecond to answer a query with error less than 0.1.

The offline computation time depends on the strategy. We break down the various steps of the offline computation in Table 3. Observe that in some cases the computation time depends on the time it takes to perform a BFS in each dataset, shown in Table 1. The offline computation may include as much as four phases:

Table 1: Summary characteristics of the collections.

Dataset	$\|V\|$	$\|E\|$	$\ell_{0.9}$	$\ell_{0.5}$	c	θ	t_{BFS}
Flickr-E	588K	11M	7	6	0.15	2.0	14s
Flickr-I	800K	18M	7	6	0.11	1.9	23s
Wikipedia	4M	49M	7	6	0.10	2.9	71s
DBLP	226K	1.4M	10	8	0.47	2.7	1.8s
Y!IM	94K	265K	22	16	0.12	3.2	<1s

$\ell_{0.9}$: effective diameter, $\ell_{0.5}$: median diameter, θ: power-law coefficient, c: clustering coefficient, t_{BFS} cpu-time in seconds of a breadth-first search.

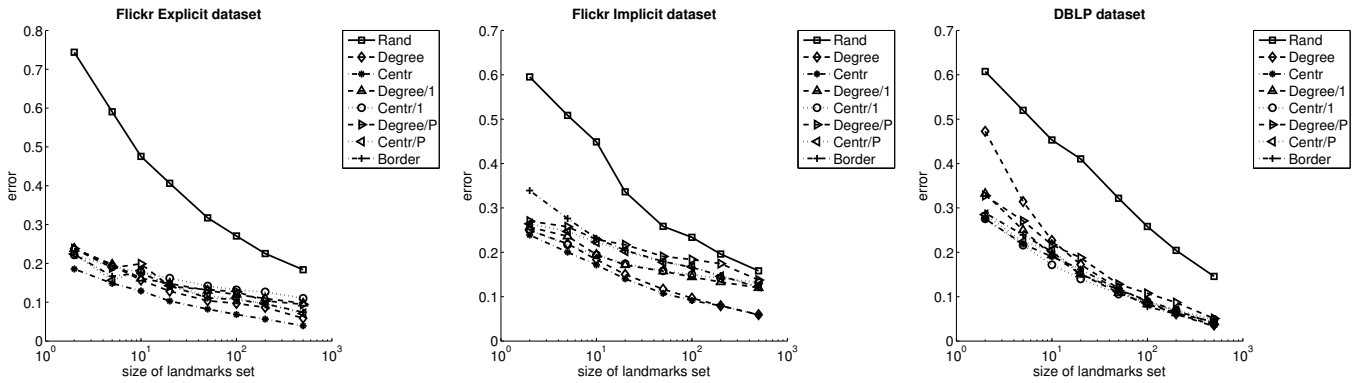

Figure 3: Error of random-pair shortest paths on three datasets.

Table 2: Summary of approximation error across datasets, using 20 landmarks (top) and 100 landmarks (bottom).

20 landmarks	Fl.-E	Fl.-I	Wiki	DBLP	Y!IM
RANDOM	0.41	0.34	0.69	0.41	0.29
DEGREE	0.13	0.15	0.35	0.17	0.37
CENTRALITY	**0.10**	**0.14**	**0.21**	0.16	0.16
DEGREE/1	0.15	0.17	0.35	0.15	0.35
CENTRALITY/1	0.16	0.17	**0.21**	0.14	**0.14**
DEGREE/P	0.14	0.22	0.40	**0.13**	0.16
CENTRALITY/P	0.14	0.20	0.22	0.15	**0.14**
BORDER/P	0.14	0.20	0.29	0.15	0.15

100 landmarks	Fl.-E	Fl.-I	Wiki	DBLP	Y!IM
RANDOM	0.27	0.23	0.61	0.26	0.14
DEGREE	0.10	0.10	0.22	0.09	0.32
CENTRALITY	**0.07**	**0.09**	**0.16**	0.09	0.11
DEGREE/1	0.12	0.14	0.23	**0.08**	0.10
CENTRALITY/1	0.13	0.15	**0.16**	0.09	0.10
DEGREE/P	0.13	0.18	0.26	0.11	0.09
CENTRALITY/P	0.11	0.17	0.16	**0.08**	0.08
BORDER/P	0.11	0.17	0.20	**0.08**	**0.07**

1. A centrality computation is required for the methods based on CENTRALITY, and it takes S BFSs in which S is the sample size of the initial seed nodes.
2. A partition of the graph is required for the methods based on partitioning */P, and in the case of the Flickr-E dataset (588K nodes, 11M edges) it takes around 30 seconds using the standard clustering method of the METIS-4.0 package [23].
3. The selection of the landmarks depends on the strategy, but it takes between 1 and 4 seconds in the Flickr-E dataset.
4. The embedding implies labelling each node in the graph with its distance to the landmarks.

The computational time during the indexing is dominated by the BFS traversals of the graph. S such traversals are necessary for performing centrality estimations in the algorithms, basically by picking S seed nodes uniformly at random and then doing a BFS from those nodes; the centrality of a node is then estimated as its average distance to the S seeds. The embedding computation always takes d BFSs. Note that this step may be parallelized.

Table 3: Indexing time. Partitioning and selecting times are expressed in wallclock seconds for $d = 100$ landmarks in the Flickr-E dataset

Method	Centrality [t_{BFS}]	Partition [sec]	Select [sec]	Embed [t_{BFS}]
RANDOM	-	-	<1	d
DEGREE	-	-	<1	d
CENTRALITY	S	-	<1	d
DEGREE/1	-	-	<1	d
CENTRALITY/1	S	-	<1	d
DEGREE/P	-	<30	4	d
CENTRALITY/P	S	<30	4	d
BORDER/P	-	<30	4	d

With respect to the memory requirements to store the index, in all of our datasets more than 99% of the pairs are at a distance of less than 63, meaning that we can safely use six bits per landmark and node to store the embeddings. With 20 landmarks in large a graph of 100 M nodes, we would use 120 bits (15 bytes) per node. Thus, around 1.5 GB of memory would be required to store all the embeddings, which is a very small memory footprint for this application. For the case that either the landmarks or the nodes are more and the index needs to resort in the disk, we store each node's embedding in a single page. Thus, we perform one page access at query-time.

5.4 Triangulation

The *triangulation* performance of random landmarks has been theoretically analyzed by Kleinberg et al. [24]. In their paper they prove bounds for the performance of the bound that support the claim that random landmarks work well in practice. In this section we provide experimental evidence that even though selecting landmarks at random makes it possible to prove bounds on the triangulation task, in all our datasets the methods we propose outperform RANDOM.

Our methods produce a smaller range within which the actual distance certainly lies in. For any pair of vertices one can measure the ratio between the best lower bound and the best upper bound that can be achieved given a set of landmarks.

Recall from Section 3.3 that using the landmarks we can provide both an upper (U) and a lower bound (L) for the actual shortest path distance. We measure the average ra-

Table 4: Summary of average ratio between L and U across datasets, using 20 landmarks (top) and 100 landmarks (bottom)

20 landmarks	Fl.-E	Fl.-I	Wiki	DBLP	Y!IM
RANDOM	0.23	0.24	0.19	0.25	0.35
DEGREE	0.32	0.27	0.24	0.33	0.24
CENTRALITY	0.31	0.28	0.24	0.31	0.38
DEGREE/1	**0.34**	**0.30**	0.24	0.35	0.26
CENTRALITY/1	0.30	0.29	0.24	0.34	0.40
DEGREE/P	**0.34**	**0.30**	0.23	0.34	**0.45**
CENTRALITY/P	0.31	**0.30**	0.24	0.33	0.41
BORDER/P	**0.34**	**0.30**	**0.25**	**0.36**	**0.44**
100 landmarks	Fl.-E	Fl.-I	Wiki	DBLP	Y!IM
RANDOM	0.32	0.31	0.28	0.36	0.50
DEGREE	0.39	0.32	**0.34**	0.44	0.28
CENTRALITY	0.38	0.33	0.32	0.41	0.44
DEGREE/1	0.40	0.37	**0.34**	**0.47**	0.50
CENTRALITY/1	0.38	0.34	0.33	0.43	0.47
DEGREE/P	**0.42**	**0.38**	**0.34**	**0.47**	**0.58**
CENTRALITY/P	0.39	0.35	0.33	0.44	0.53
BORDER/P	**0.42**	**0.38**	**0.34**	0.46	**0.58**

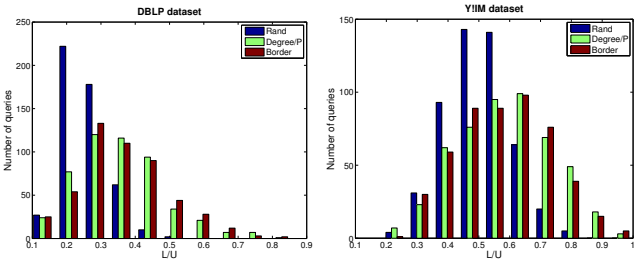

Figure 4: Distributions of L/U ratio

tio (L/U) over all queries and present the results in Table 4. Observe that BORDER/P and DEGREE/P outperform RANDOM in all datasets. They also outperform the rest of the strategies with smaller margins. We note that one needs to use one or two orders of magnitude more landmarks in RANDOM to achieve the accuracy of BORDER and DEGREE/P.

A more detailed view is presented in Figure 4. We plot histograms that reflect the distribution of the L/U ratio for 20 seeds in DBLP and for 100 seeds in Y!IM. Observe that the volume of the distributions of methods BORDER and DEGREE/P are similar and that they are both clearly closer to the unit than the respective distribution of RANDOM.

5.5 Comparison with exact methods

We compare our method with state-of-the-art algorithms (ALT) for computing exact point-to-point shortest paths, as described by Goldberg and Harrelson [16, 17]. We note that our methods can be considered as fair competitors with ALT, since the ALT methods not only compute exact distances, but they also compute the shortest path itself and not only its length. Nevertheless, our comparison can be seen as an illustration on how much one can gain if only the length of the shortest path is needed and if one is willing to tolerate small approximation errors.

In brief, ALT methods combine bidirectional Dijkstra traversal, with A* and landmark-based lower bounds. We implemented[2] Pohl's [27] and Ikeda's [22] algorithm. Landmarks were chosen using Goldberg's *farthest* heuristic [17]. Below we report results only for Ikeda's algorithm, because it outperformed Pohl's algorithm in all of our datasets; the same result is reported in [16].

[2]Code from [16, 17] is proprietary and not publicly available

Table 5: Comparison with ALT methods.

Ours (10%)	Fl.-E	Fl.-I	Wiki	DBLP	Y!IM
Method	CENT	CENT	CENT/P	BORD/P	BORD/P
Landmarks used	20	100	500	50	50
Nodes visited	1	1	1	1	1
Operations	20	100	500	50	50
CPU ticks	2	10	50	5	5
ALT (exact)	Fl.-E	Fl.-I	Wiki	DBLP	Y!IM
Method	Ikeda	Ikeda	Ikeda	Ikeda	Ikeda
Landmarks used	8	4	4	8	4
Nodes visited	7245	10337	19616	2458	2162
Operations	56502	41349	78647	19666	8648
CPU ticks	7062	10519	25868	1536	1856

The results are summarized in Table 5. For our methods, we use as many landmarks are needed to bring the error below 0.1. For ALT, since it is an exact method, we use the number of landmarks that gives the best performance. In the rows labeled OPERATIONS in Table 5 we report the number of arithmetic operations during the LB/UB computations. This measure gives a large advantage to ALT methods since it disregards costs involving hashing visited nodes, priority queue maintenance and most importantly random accesses to nodes. We also present the average number of CPU ticks, noting that this measure is implementation dependent: our implementation is main memory based and uses the C++/STL priority queue template.

Table 5 confirms that our techniques outperform the state-of-the-art for exact Point-to-point SP by several orders of magnitude according to all measures while suffering a very small loss in accuracy in all five datasets.

6. APPLICATION TO SOCIAL SEARCH

In this section we describe an application of our method to network-aware search. In network-aware search the results of a search are nodes in a graph, and the originating query is represented by a *context node* in the graph. The context node may represent the user issuing the query, or a document the user was browsing when she issued the query. Nodes that match the query are ranked using a ranking function that considers, among other factors, their connection with the context node; for instance, the ranking function may favor the results that are topologically close to the context node.

6.1 Problem definition

There are a number of use cases where social search may be helpful. For instance, a user may be searching on a social networking site for a person which she remembers only by the first name. There might be potentially thousands of matching people, but friends-of-friends would be ranked higher since they are more likely to be acquaintances of the query-issuer. As another application consider a user searching for books, music or movies of a certain genre: items favorited by her friends or friends-of-friends are more likely to be interesting for her. Yet another application is context-aware search, where a user is reading a page with a search form on it (e.g. Wikipedia) and enters a query. Pages linked

to by the original page (or close by in terms of clicks) should be presented first.

The problem we consider was defined by Vieira et al. [36]. Given a source vertex and a set of matching-vertices that satisfy the query (which are provided by an inverted index), rank the matching vertices according to their shortest path distance from the source vertex.

6.2 Evaluation method

To evaluate the effectiveness of our methods for search, we need a method for generating search tasks, as well as a performance metric. For the latter we use *precision@k* denoted by p@k; this is the size of the intersection between the top k elements returned by our method using approximate distances, and the top k elements in the result set X, normalized by k. Given that there might be elements of X tied by distance, we extend X to include the elements at the same distance as the k-th element on X, as any of those elements can be considered a top-k result to the query.

We point out that while this evaluation method emulates a hypothetical ranking function that uses only the distances, in most practical settings the distance between two items should be a *component* of the ranking function, not a replacement for it.

To generate queries and select the matching results we consider that each element in the graphs has a set of *tags* or keywords associated to it. In the case of the Flickr datasets (both explicit and implicit graphs) tags are naturally provided by users; a user has a tag if she has tagged at least one photo with that tag . In the case of the Yahoo! Instant Messaging graph, we cross the information with the items users have rated in Yahoo! Movies (we have been provided with an anonymized dataset in which this information is already joined), so the tags of a user are the movies she has rated. In the case of the Wikipedia dataset, the tags are the words the pages contain. We select words that are neither uncommon nor trivially common: we pick words that have document frequency between $1K$ and $10K$. In the case of DBLP, we use artificial tags. We create random tags and assign them to 100 random users in the graph.

6.3 Experimental results

Table 6 summarizes the precision at 5 of the strategies for 20 and 100 landmarks. The RANDOM technique is outperformed in all datasets, by large margins.

Figure 5 shows how the precision increases by adding more landmarks. The x-axis is logarithmic so it is clear that returns are diminishing. In any case a few tens of landmarks are enough for Flickr and a few hundred landmarks for the other datasets. Results in Table 6 agree with those in Table 2 and similar conclusions hold. The constrained CENTRAL/1 and the partitioning-based BORDER/P yield consistently good results across all datasets. We note that the results for other precision measures, such as p@1, p@2,p@10 and p@20, are qualitatively similar to p@5.

As a general observation, the number of landmarks necessary for a good approximation depends much more on the graph structure than on the graph size. For instance, despite Y!IM being the smallest graph, it is also the one that exhibits the larger range of distances, as shown in Figure 2. As expected, on this graph the search task requires more landmarks to obtain a high precision than on the other graphs.

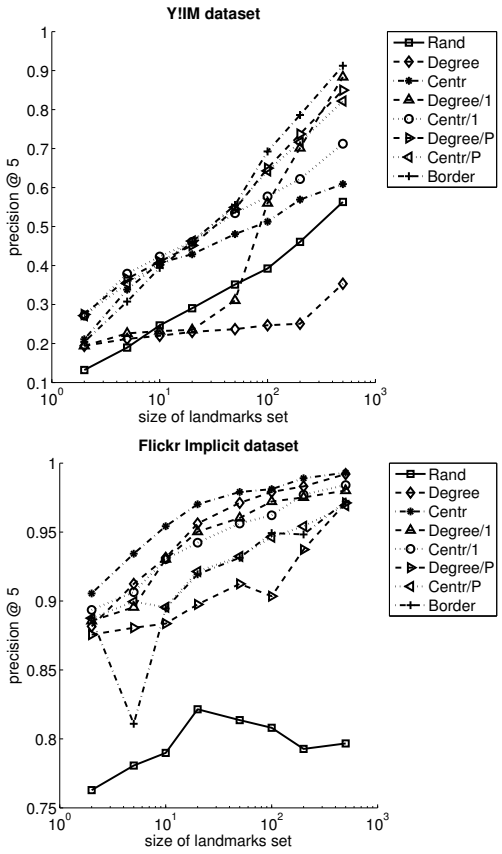

Figure 5: Precision at 5 for the social search task

Table 6: Summary of precision at 5 across datasets, using 20 and 100 landmarks

20 landmarks	Fl.-E	Fl.-I	Wiki	DBLP	Y!IM
RANDOM	0.84	0.82	0.42	0.60	0.29
DEGREE	**0.98**	0.96	0.46	0.77	0.23
CENTRAL	**0.98**	**0.97**	**0.58**	0.77	0.43
DEGREE/1	0.97	0.95	0.46	0.78	0.23
CENTRAL/1	0.97	0.94	**0.58**	**0.80**	**0.46**
DEGREE/P	0.97	0.90	0.50	0.78	0.45
CENTRAL/P	0.97	0.92	0.57	0.78	**0.46**
BORDER/P	**0.98**	0.92	0.53	0.79	**0.46**
100 landmarks	Fl.-E	Fl.-I	Wiki	DBLP	Y!IM
RANDOM	0.86	0.80	0.59	0.65	0.39
DEGREE	**0.99**	**0.98**	0.67	0.88	0.25
CENTRAL	**0.99**	**0.98**	0.69	0.87	0.51
DEGREE/1	0.98	0.97	0.67	**0.92**	0.56
CENTRAL/1	0.98	0.96	0.68	0.89	0.58
DEGREE/P	0.98	0.90	0.65	0.88	0.65
CENTRAL/P	0.97	0.95	**0.71**	0.89	0.64
BORDER	**0.99**	0.95	0.68	0.91	**0.69**

6.4 External memory implementation

In this section we consider the case that the distances of the graph nodes to the landmarks do not fit in the main memory and need to be stored on disk. Recall that the query returns an answer set of relevant nodes. These nodes are only known at query-time. Depending on the query selectivity s (i.e., the size of the answer set) we may follow different strategies: if s is small, the methodology of Sec-

tion 5.3 applies; if the index is stored in external memory, we need to perform s page accesses per query.

If s is large then we follow a different strategy. First, observe that for large s it is meaningful to retrieve the top-k for some $k \ll s$. To that end, we use ideas similar to the *Threshold Algorithm* (TA) algorithm, introduced in the seminal work of Fagin et al. [12]. The application of TA algorithms is as follows: nodes and their distance to a given landmark are stored in ascending order. There is one such list per landmark. We visit the lists sequentially using only *sorted access* until we get the top-k relevant answers. There are two important observations. First, during query-processing, a list (i.e., landmark) may be pruned as a whole using the greatest distance in the current top-k set as a threshold. Second, this approach employs mainly sequential access in disk which is much faster than random access.

As a final remark, consider a system that implements the social search task. Recall that Dijkstra/BFS will very quickly retrieve nodes with small distance from the query-node, but its efficiency degrades very fast in social-network like graphs. Contrary to that, embedding based methods are very efficient, especially for low selectivity queries. Thus a hybrid approach which combines both would optimize efficiency; its details are not in the scope of this work.

7. CONCLUSIONS AND FUTURE WORK

Motivated by applications such as context-aware web search and socially-sensitive search in social networks, we studied how to do fast and accurate distance estimation on real-world massive graphs using landmarks. We characterized the problem of optimal landmark selection and proved that it is **NP**-hard. We described several strategies for landmark selection which outperform the current approximate standard RANDOM and the state-of-the-art exact techniques by large margins according to our extensive experimentation. In the simplest class of strategies, CENTRALITY appears to be much more robust than DEGREE. Among the more elaborate strategies, the ones based on partitioning, in particular BORDER/P, exhibit consistent savings in computational cost across all datasets. When applied to the task of context-aware search these methods yield results of high precision.

In our on-going work we are studying methods for an effective synergy of upper and lower bounds. We also plan to investigate the issue of dynamic data structures and how to deal with frequent network updates. A key aspect is to investigate methods to provide estimates in grams for distance functions other than the shortest path distance, which could also be used as primitives in context and/or social aware search tasks.

8. REFERENCES

[1] I. Abraham, Y. Bartal, H. Chan, K. Dhamdhere, A. Gupta, J. Kleinberg, O. Neiman, and A. Slivkins. Metric embeddings with relaxed guarantees. In *FOCS 2005*.

[2] S. Amer-Yahia, M. Benedikt, L. V. Lakshmanan, and J. Stoyanovic. Efficient network-aware search in collaborative tagging sites. In *VLDB 2008*.

[3] V. Athitsos, P. Papapetrou, M. Potamias, G. Kollios, and D. Gunopulos. Approximate embedding-based subsequence matching of time series. In *SIGMOD 2008*.

[4] D. Bader, S. Kintali, K. Madduri, and M. Mihail. Approximating betweenness centrality. In *WAW 2007*.

[5] Baeza and Ribeiro. *Modern Information Retrieval. ACM Press / Addison-Wesley, 1999.*

[6] J. Bourgain. On Lipschitz embedding of finite metric spaces in Hilbert space. *Israel Journal of Mathematics*, 52(1):46–52, March 1985.

[7] U. Brandes. A faster algorithm for betweenness centrality. *Journal of Mathematical Sociology*, 2001.

[8] V. Chvatal. A greedy heuristic for the set-covering problem. *Mathematics of Operations Research*, 1979.

[9] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms, 2nd Edition.* The MIT Press, 2001.

[10] F. Dabek, R. Cox, F. Kaashoek, and R. Morris. Vivaldi: a decentralized network coordinate system. In *SIGCOMM 2004*.

[11] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1959.

[12] R. Fagin, A. Lotem, and M. Naor Optimal aggregation algorithms for middleware. J. Comput. Syst. Sci., 2003.

[13] R. W. Floyd. Algorithm 97: Shortest path. *Commun. ACM*, 5(6), June 1962.

[14] D. Fogaras, and B. Racz Towards Scaling Fully Personalized PageRank. *Algorithms and Models for the Web-Graph*, pp. 105-117, 2004.

[15] L. Freeman. A set of measures of centrality based on betweenness. *Sociometry*, 40(1):35–41, 1977.

[16] A. Goldberg, H. Kaplan, and R. Werneck. Reach for A^*: Efficient point-to-point shortest path algorithms. Tech. Rep. MSR-TR-2005-132, October 2005.

[17] A. Goldberg and C. Harrelson. Computing the shortest path: A* search meets graph theory. In *SODA 2005*.

[18] A. Goyal, F. Bonchi, and L. Lakshmanan. Discovering leaders from community actions. In *CIKM 2008*.

[19] B. M. Hill. A simple general approach to inference about the tail of a distribution. *Annals of Stat.*, 1975.

[20] G. Hjaltason and H. Samet. Properties of embedding methods for similarity searching in metric spaces. *IEEE Trans. Pattern Anal. Mach. Intel.*, 25(5):530–549, 2003.

[21] W. Hoeffding. Probability inequalities for sums of bounded random variables. Journal of the American Statistical Association 58(301):13–30, 1963.

[22] T. Ikeda, M.-Y. Hsu, H. Imai, S. Nishimura, H. Shimoura, T. Hashimoto, K. Tenmoku, and K. Mitoh. A Fast Alogrithm for Finding Better Routes by AI Search Techniques. In *IEEE Vehicle Navigation and Information Systems Conference*, 1994.

[23] G. Karypis and V. Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on Scientific Comp.* 20(1):359–392, 1999.

[24] J. Kleinberg, A. Slivkins, and T. Wexler. Triangulation and embedding using small sets of beacons. In *FOCS 2004*.

[25] HP. Kriegel, P. Kroger, M. Renz, and T. Schmidt. Vivaldi: a decentralized network coordinate system. In *SIGCOMM 2004*.

[26] E. Ng and H. Zhang. Predicting internet network distance with coordiantes-based approaches. In *INFOCOM 2001*.

[27] I. Pohl. Bi-directional Search. In Machine Intelligence, vol. 6, Edinburgh University Press, 1971, pp. 127Ű-140.

[28] M. J. Rattigan, M. Maier, and D. Jensen. Using structure indices for efficient approximation of network properties. In *KDD 2006*.

[29] H. Samet, J. Sankaranarayanan, and H. Alborzi. Scalable network distance browsing in spatial databases. In *SIGMOD'08*.

[30] P. Singla and M. Richardson. Yes, there is a correlation: from social networks to personal behavior on the web. In *WWW'08*.

[31] L. Tang and M. Crovella. Virtual landmarks for the internet. In *IMC 2003*.

[32] M. Thorup and U. Zwick. Approximate distance oracles. In *ACM Symp. on Theory of Computing*, 2001.

[33] H. Tong, C. Faloutsos and J-Y.Pan. Fast random walk with restart and its applications In *ICDM*, 2006.

[34] A. Ukkonen, C. Castillo, D. Donato, and A. Gionis. Searching the wikipedia with contextual information. In *CIKM*, 2008.

[35] J. Venkateswaran, D. Lachwani, T. Kahveci, and C. Jermaine. Reference-based indexing of sequence databases. In *VLDB 2006*.

[36] M. V. Vieira, B. M. Fonseca, R. Damazio, P. B. Golgher, D. de Castro Reis, and B. Ribeiro-Neto. Efficient search ranking in social networks. In *CIKM 2007*.

[37] Y. Xiao, W. Wu, J. Pei, W. Wang, and Z. He. Efficiently Indexing Shortest Paths by Exploiting Symmetry in Graphs. In *EDBT 2009*

[38] U. Zwick. Exact and approximate distances in graphs — a survey. *LNCS*, 2161, 2001.