

Fast shortest path distance estimation in large networks

Michalis Potamias^{*}
Boston University
mp@cs.bu.edu

Carlos Castillo
Yahoo! Research, Barcelona
chato@yahoo-inc.com

Francesco Bonchi
Yahoo! Research, Barcelona
bonchi@yahoo-inc.com

Aristides Gionis
Yahoo! Research, Barcelona
gionis@yahoo-inc.com

ABSTRACT

We study the problem of preprocessing a large graph so that point-to-point shortest-path queries can be answered very fast. Computing shortest paths is a well studied problem, but exact algorithms do not scale to huge graphs encountered on the web, social networks, and other applications.

In this paper we focus on approximate methods for distance estimation, in particular using *landmark-based distance indexing*. This approach involves selecting a subset of nodes as landmarks and computing (offline) the distances from each node in the graph to those landmarks. At run-time, when the distance between a pair of nodes is needed, we can estimate it quickly by combining the precomputed distances of the two nodes to the landmarks.

We prove that selecting the *optimal* set of landmarks is an NP-hard problem, and thus heuristic solutions need to be employed. Given a budget of memory for the index, which translates directly into a budget of landmarks, different landmark selection strategies can yield dramatically different results in terms of accuracy. A number of simple methods that scale well to large graphs are therefore developed and experimentally compared. The simplest methods choose central nodes of the graph, while the more elaborate ones select central nodes that are also far away from one another. The efficiency of the suggested techniques is tested experimentally using five different real world graphs with millions of edges; for a given accuracy, they require as much as 250 times less space than the current approach in the literature which considers selecting landmarks at random.

Finally, we study applications of our method in two problems arising naturally in large-scale networks, namely, social search and community detection.

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

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, or instant messaging systems nowadays count hundreds of millions of users that are active on a daily basis. For graphs of this size even the algorithmic problems that are seemingly simple become challenging tasks in practice.

One basic operation in networks is to measure how close an entity 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 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 (see, e.g., [41]). 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 [37]. This *socially sensitive* search model has been suggested as part of the social network search experience [2].

Although computing shortest paths is a well studied problem, exact algorithms do not scale well to nowadays real-world massive networks. A full breadth-first search (BFS) traversal of a graph of 5M nodes and 50M edges takes roughly a minute in a standard modern desktop computer. 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 with fast estimates of the actual distance in very short time (i.e., milliseconds). The offline step consists of choosing a subset of nodes as *landmarks* (also referred to as *reference objects*) and computing distances from every node to them. Such precomputed information is often referred to as an *embedding*.

Our contribution. In this paper we present an extensive analysis of various heuristics for selecting landmarks. We devise and experimentally compare more than 30 heuristics 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 landmarks selection strategies for shortest path computation in large networks. Our main contributions can be summarized as follows:

- We define the problem of optimal landmark selection in a graph and prove that it is **NP**-hard (Section 3).
- We suggest simple and intuitive heuristics 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 datasets (Section 5).
- We apply our methods to search in four social networks and a webgraph, showing high precision and low error in the problem of finding the closest nodes in the graph that match a given query (Section 6).
- We suggest that the precomputed embeddings can be used for fast and meaningful graph partitioning (Section 7).

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.

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 [12]. 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 [14] 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.

Goldberg et al. [20, 21] address the problem of point-to-point shortest path (PPSP) and 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 [20, 21] since we are interested only on the length of a shortest path, not the path itself. Recently, Xiao et al have exploit graph symmetry to obtain speed-ups for PPSP queries over simple BFS traversals [45]. Our algorithms work only on the precomputed node-to-landmark-distances and do not perform any Dijkstra-type computation at query-time.

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 [39] 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 straightforward exact solution: compute all shortest paths and store them. For large graphs such as the ones considered in this paper, even if one could afford the computational cost of APSP, the space cost would be asymptotically greater than the one needed to store the graph itself.

For $k = 2$ the estimate may be three times more than the actual distance. In large real-world graphs this bound is already problematic due to the *small-world phenomenon*; in a scale-free network such as the explicit 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. A survey on exact and approximate distances in graphs can be found in [46].

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 employing a more efficient distance function such as an L_p norm. Several methods have been proposed to embed a space into a Euclidean space [7, 24]. There have been attempts to optimize the selection of reference objects for such a setting [3, 42]. Other dimensionality reduction techniques are also widely used. Principal Component Analysis (PCA) and related techniques are used to project high dimensional data in a lower dimensional space, while maximizing the preserved variance of the data. As described in 3, our distance function incorporates a \min , and thus PCA techniques cannot be applied to further reduce the dimensionality of our embedding.

Landmarks have been already been used for internet measurements [11, 34, 38]. Rattigan et al. [35] combine their *zone* approach with the landmark technique in order to measure shortest paths in general graphs and finally to estimate centrality measures. Our work may be combined with their techniques but this combination is out of the scope of this paper. Kleinberg et al. discuss the problem of approximating network distances in real networks (not necessarily shortest paths) via embeddings using a small set of *beacons* (i.e., landmarks) [28]. Of most interest is the fact that they introduce in their analysis the notion of *slack*, as a quantity 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 heuristics work much better than the random. Abraham et al. generalize the metric embedding with slack [1]. On another perspective, computing shortest paths in spatial networks has also attracted interest recently [30, 36]; our work is different since we focus on graphs that exhibit complex social network or web-graph behavior.

Applications. Several measures have been introduced to measure the *centrality* of a vertex. Our work is tightly connected to these notions. Betweenness centrality measures the amount of shortest paths passing from a vertex while closeness centrality measures the average distance of a vertex to any other vertex in the network [17]. Brandes gave the best known algorithm to compute the exact betweenness centrality of all vertices by adapting the APSP Dijkstra algorithm [8]. The algorithm runs in $O(nm + n^2 \log n)$ time, which is prohibitive for large graphs. Bader et al. gave a sampling-based approximation algorithm [5] and showed that centrality is easier to approximate for central nodes. In our work we use closeness centrality as a heuristic in choosing central points as landmarks in the graph.

Fast PPSP computation is becoming very relevant to Information Retrieval. Socially sensitive search in social networks and location-aware search are attracting remarkable interest in the information retrieval community [6]. It has been reported that people who chat with each other are more likely to share interests [37]. 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. [43]. Their work is also based on landmarks, but their landmarks are chosen randomly. Recently Amer-Yahia et al. consider network-aware search in collaborative tagging sites [2]. In the web-search filed, Kraft et al. suggest that a query in web-search can be enriched with context from its source, i.e., the page the user was browsing at the time of the query initiation [29]. Recently Ukkonen et al. used this idea as a component of the ranking function in searching within Wikipedia [41]. In the same context, research interest has been attracted into approximating other graph proximity functions that are based on random walks, such as personalized pagerank and random walk with restart [15, 40]. The growing interest in involving context and/or social connections in searching 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 deadlines to meet, in the order of hundreds of milliseconds. Our methods can provide accurate results within these deadlines.

Finally, community detection has received interest in physics and sociology [16, 19, 33, 44]. More recently, with the increasing availability of very large graphs, computer scientists have started studying community detection in the web [18, 32], and in social networks [4, 26, 31]. In Section 7 we suggest that, when landmarks are chosen appropriately, node embeddings may be used as features for clustering; thus the community detection task may be reduced to standard clustering on relational data.

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 $\pi_{s,t} = \langle s, u_1, u_2, \dots, u_{\ell-1}, t \rangle$ to be a path of distance $|\pi_{s,t}| = \ell$ between s and t , if $\{u_1, \dots, u_{\ell}\} \subseteq V$ and $\{(s, u_1), (u_1, u_2), \dots, (u_{\ell-1}, t)\} \subseteq E$,

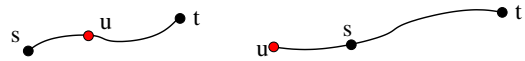


Figure 1: Illustration of the cases for obtaining tight upper bounds (left) and tight lower bounds (right) as provided by Observations 1 and 2

and let $\Pi_{s,t}$ be the set of all paths from s to t . Accordingly, let $d_G(s, t)$ be the distance corresponding to the shortest path between any two vertices $s, t \in V$. 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 $\text{SP}_{s,t}$ be the set of paths whose length is equal to $d_G(s, t)$.

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 $D = \langle u_1, u_2, \dots, u_d \rangle$ 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 d -dimensional vector $\phi(v)$:

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

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 $\pi_{s,t} \in \text{SP}_{s,t}$ 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 $\pi_{s,u} \in \text{SP}_{s,u}$ so that $t \in \pi_{s,u}$, or there exists a path $\pi_{t,u} \in \text{SP}_{t,u}$ 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 about 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 in a shortest path that 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* [17].

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 [8]. They extend

well-known all-pairs-shortest-paths algorithms [10]. The time cost $O(nm)$ for unweighted graphs and $O(nm + n^2 \log n)$ for weighted graphs. Additionally, Bader et al. [5] 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 [8].

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. \square

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 t . Then by solving the SET-COVER problem on (U, \mathcal{S}) by the greedy algorithm [9] 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 heuristics of the next section are motivated by the observations made in this section regarding properties of good landmarks.

4. LANDMARK-SELECTION HEURISTICS

The baseline heuristic, which has been used by many researchers in the literature, is to select landmarks at random

[43, 28, 38]. The heuristics 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.

For selecting a set of landmarks that cover many different pairs of nodes, we want to *spread* the landmarks at different positions of the graph. To capture this intuition we propose two modifications of our heuristics: (i) a *constrained* variant, where we do not select landmarks that are nearby, and (ii) a *partitioning* variant, where we first partition the graph and then select landmarks from different partitions.

4.1 Basic heuristics

RANDOM: The baseline landmark-selection heuristic 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 heuristic 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 [25] to show that we can obtain arbitrarily good approximation to centrality by sampling a constant number of seeds.

4.2 Constrained heuristics

Our goal is to cover as many pairs as possible. Using a basic heuristic 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 heuristics depends on a depth parameter h . We first rank the nodes according to some heuristic (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 heuristics by DEGREE/ h and CENTRALITY/ h .

For all the experiments we report 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 heuristics

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 [27]) and then select landmarks from the different partitions. We suggest the following partitioning-based heuristics.

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:

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

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 heuristics that combine both partitioning and constrained heuristics.

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 heuristics we discuss above are optimized to give good upper-bound landmarks, by selecting central nodes in the graph, and they perform very 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 the following algorithm: (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

This section presents experimental results in terms of approximation error and running time, for five datasets.

5.1 Datasets

In order to demonstrate the robustness of our methods and to show their performance on realistic data, 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 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, and 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.

The sample of Flickr we use has 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. We infer implicit contacts between Flickr users using *reciprocal comments* 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 coauthor 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 users who are active also in Yahoo! Movies. Goyal et al. describe this dataset in detail [22].

Wikipedia hyperlinks. Apart from the previous four datasets, which are social 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 Zipfian distribution in which the probability of having degree x is proportional to $x^{-\theta}$; the parameter θ fitted using Hill’s estimator [23] 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. For this experiment we randomly choose 500 random pairs of nodes. In the case of the RANDOM selection heuristic 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.

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 shows representative results for two datasets. Observe that using two landmarks chosen with the CENTRALITY heuristic 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 heuristics 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,

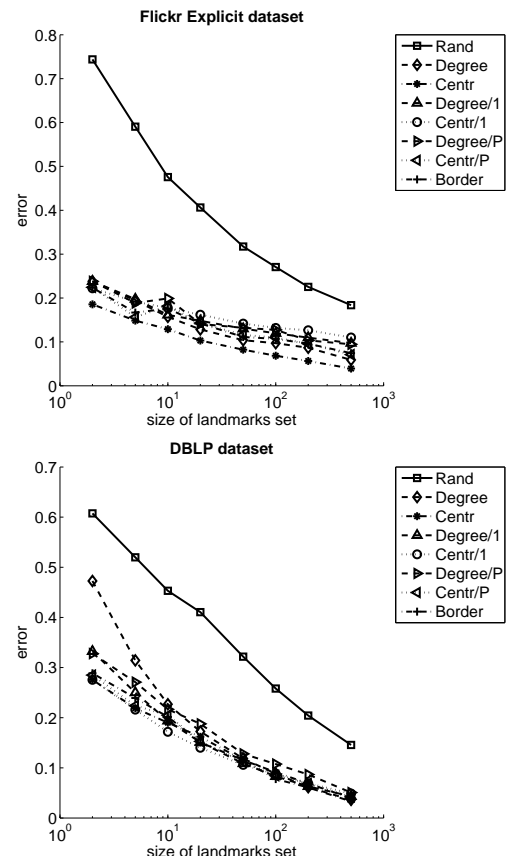


Figure 3: Error of random-pair shortest paths

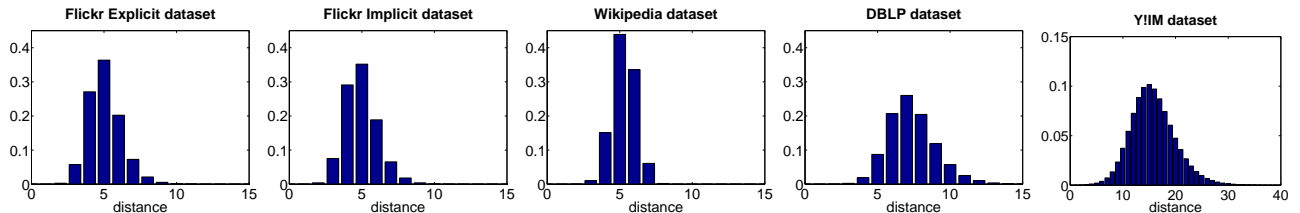


Figure 2: Distributions of distances in our datasets.

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

20 landmarks	Fl.-E	Fl.-I	Wiki	DBLP	YIM
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	YIM
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

but sometimes does not perform well, as in the case of the YIM 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 Boost and STL Libraries¹. 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. This time is at most some milliseconds in practice. A comparison with the BFS times presented in table 1 and the fact that the losses in accuracy are as small as 10%, illustrates the power of our methods.

The offline computation time depends on the heuristic. 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:

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.

¹Code implementing the heuristics described in this paper will be available at publication time in <http://barcelona.research.yahoo.com/distidx/>.

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

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.²
3. The selection of the landmarks depends on the heuristic, 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. These traversals can be sped-up by doing the traversals in parallel in several machines. Observe that the exact solution which requires a BFS/Dijkstra traversal of the graph, cannot be straightforwardly parallelized.

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.

²<http://glaros.dtc.umn.edu/gkhome/metis/metis/overview>

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

5.4 Triangulation

In this subsection we demonstrate the surprising result that our methods outperform the random landmarks selection in the *triangulation* task. In other words 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. The *triangulation* performance of random landmarks has been studied by Kleinberg et al. [28]. In their paper they prove bounds for the performance of the bound that support the claim that random landmarks work well in practice. Our methods outperform RANDOM in this task in all 5 large networks by large margins.

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. The closer these two bounds are expected to be to each other the better the landmark selection. We measure the average ratio (L/U) over all queries and present the results in Table ???. Observe that the bounds obtained from the heuristics are much tighter than RANDOM across all datasets, with the exception of DEGREE in Y!IM. Also, notice that BORDER and DEGREE/P outperform RANDOM by large margins in all datasets. They also outperform the rest of the heuristics but with smaller margins. We note that one needs to use between one and two orders of magnitude more RANDOM landmarks in order to achieve the quality of BORDER and DEGREE/P in the triangulation task.

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 favored 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 has been defined by Vieira et al. [43]. 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. A concrete definition is:

DEFINITION 2. *Given a graph $G(V, E)$, a source vertex $q \in V$ and a set $X = \{x_1, x_2, \dots, x_{|X|}\} \subseteq V$ that satisfies some query introduced by vertex q , compute a permutation $\bar{\pi} = \langle \pi_1, \pi_2, \dots, \pi_{|X|} \rangle$ of the items in the set $\{1, 2, \dots, |X|\}$, so that for the ranking $\langle x_{\pi_1}, x_{\pi_2}, \dots, x_{\pi_{|X|}} \rangle$ it is true that $\nexists \pi_i, \pi_j | i < j \wedge d_G(q, x_{\pi_i}) > d_G(q, x_{\pi_j})$, where $d_G(x, y)$ is the graph-distance of vertices x, y .*

In practice, a query is a tuple $\langle q, t \rangle$ where q is the asking vertex and t is a set of keywords or tags that must be matched against all the elements of the collection to select the candidate set X . Note that both the source vertex q and the set of relevant items X are only known at runtime.

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, filtering out tags that have been used in less than 100 photos. 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

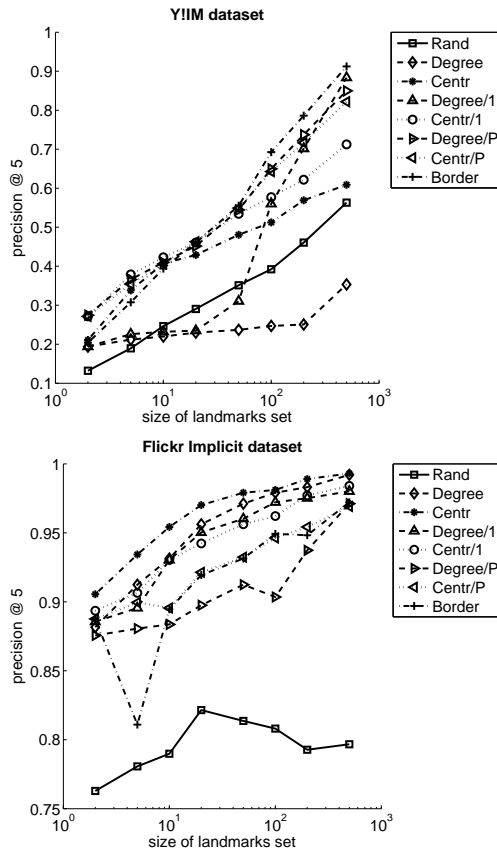


Figure 4: Precision at 5 for the social search task

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 the 650 words that have frequency between 1K and 10K. In the case of DBLP, we use artificial tags. We create 201 tags and we assign it to 100 random users in the graph.

6.3 Experimental results

Table 5 summarizes the precision at 5 of the heuristics for 20 and 100 landmarks. The RANDOM technique is clearly outperformed in all datasets.

Figure 4 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. Table 5 is quite consistent with Table 2 and similar conclusions hold. Among the simple strategies CENTRALITY is better and DEGREE performs poorly in the Y!IM dataset. The constrained CENTRAL/1 and the partitioning-based BORDER/P yield good results across all datasets. We note that the results for all other interesting precision measures, namely $p@1$, $p@2$, $p@10$ 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

Table 5: 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

graph structure than on the graph size. For instance, despite Y!IM being the smallest graph, the search task requires more landmarks to obtain a high precision than for the other graphs.

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 Section 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. [13]. 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. APPLICATION TO COMMUNITY DETECTION

In this section we describe a preliminary investigation on the applicability of landmark-based embeddings to the task of clustering the nodes of a graph. The underlying intuition is that, when landmarks are chosen appropriately, node embeddings may contain rich information about the node position within the graph. By using the resulting embeddings and adopting a standard Euclidean distance between embeddings, we reduce the community detection task to a standard clustering on relational data. We put in practice the above intuition on the DBLP dataset as follows.

First, we use the DEGREE/1 method to generate a set of 100 landmarks and the corresponding node embeddings. Then we use them as input to standard k -means ($k = 50$).

In order to assess the meaningfulness of the obtained author-clusters we inspect the area of publication of the clusters. In particular, we compute the *salient journals* inside each cluster as follows. For each journal paper we give $1/k$ points to each of its k co-authors. For each cluster and for each journal we take the sum of the points for the given journal over all authors in the cluster.

Next we find if a cluster has more than 20% of the total points associated to a given journal, i.e., the cluster of authors contributes to more than 20% of a given journal history of publications. In Table 6 we list the journals that get more points inside each of those clusters. Some clusters disappear as they may not reach the 20% threshold for any journal. We report some interesting clusters that are discovered.

It is worth noting how two different communities of the large DB community arise: one mixed with artificial intelligence (Cluster 35), and one mixed with data mining and algorithms (Cluster 10).

The results are promising: recall that we cluster the nodes (i.e., the authors), not the journals. Moreover, the clusters are obtained using only the embeddings and no additional information. In our future work we plan to further investigate this landmark-based approach to community discovery and compare with other methods.

8. 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. In particular we focused on approximate methods based on landmarks. We characterized the problem of optimal landmark selection and proved that it is NP-hard. Thus we described several heuristics for landmark selection which outperform the current standard RANDOM by a large margin according to our extensive experimentation. In the simplest class of heuristics, CENTRALITY appears to be much more robust than DEGREE. Among the more elaborate heuristics, the ones based on partitioning, in particular BORDER/P, are the most promising. When applied to the task of context-aware search in four social networks and a webgraph, these methods show high precision and low error. Finally, the embedding may provide with meaningful graph partitioning using standard clustering methods: coherent communities of authors were found in the DBLP dataset.

In our on-going work we are studying methods for an effective synergy of upper and lower bounds. The idea of using landmark-based embeddings for community detection is also worth a deeper investigation. Of most interest is to investigate the means to provide estimates for more distance

Table 6: Example clusters obtained in DBLP.

Cluster 1 - 8693 authors - Graphs, Combinatorics, Discr. Math.	
Contrib. %	Journal
0.32	Algorithmic Operations Research
0.25	Graphs and Combinatorics
0.23	Journal of Graph Theory
0.22	Discrete Applied Mathematics
0.22	Combinatorics, Probability & Computing
0.21	Discrete Mathematics
0.21	Journal of Discrete Algorithms
0.21	Electronic Colloquium on Computational Complexity

Cluster 10 - 5375 authors - DB, KDD, Algorithms	
Contrib. %	Journal
0.67	SIGMOD Digital Review
0.38	IEEE Data(base) Engineering Bulletin
0.35	ACM Transactions on KDD (TKDD)
0.33	Internet Mathematics
0.33	Theory of Computing
0.33	SIGMOD Record
0.32	The VLDB Journal
0.31	Advances in Computing Research
0.31	Algorithmica
0.30	ACM Transactions on Database Systems (TODS)
0.30	Data Mining and Knowledge Discovery
0.30	ACM Transactions on Algorithms

Cluster 16 - 6012 authors - Imaging, Vision, Virtual Reality	
Contrib. %	Journal
0.25	Journal of Mathematical Imaging and Vision
0.22	International Journal of Virtual Reality

Cluster 21 - 5970 authors - Algorithms	
Contrib. %	Journal
0.36	Computational Geometry: Theory and Applications
0.34	Theory of Computing
0.33	ACM Transactions on Algorithms
0.31	Journal of Graph Algorithms and Applications
0.31	Random Structures and Algorithms
0.30	Electronic Colloquium on Computational Complexity
0.30	Journal of Automata, Languages and Combinatorics
0.30	Computational Complexity
0.29	Combinatorics, Probability & Computing
0.28	Combinatorica
0.27	Discrete & Computational Geometry
0.27	ACM Journal of Experimental Algorithms (JEA)
0.27	SIAM Journal on Computing
0.27	Journal of Algorithms
0.26	Algorithmica
0.26	Internet Mathematics

Cluster 35 - 6938 authors - DB, Artificial Intelligence	
Contrib. %	Journal
0.34	Journal of Web Semantics
0.27	SIGMOD Record
0.26	The VLDB Journal
0.24	Autonomous Agents and Multi-Agent Systems
0.23	IEEE Data(base) Engineering Bulletin
0.23	ACM Transactions on the Web
0.23	AI Magazine
0.23	Journal of Artificial Intelligence Research (JAIR)
0.22	SIGMOD Digital Review
0.21	Intelligence
0.21	ACM Transactions on Database Systems (TODS)
0.21	Artificial Intelligence

Cluster 44 - 8230 authors - Architecture, Parallelism, Concurrency	
Contrib. %	Journal
0.35	Journal of Instruction-Level Parallelism
0.34	ACM Trans. on Arch. and Code Opt. (TACO)
0.32	Computer Architecture Letters
0.24	ACM Letters on Progr. Lang. and Syst. (LOPLAS)
0.22	International Journal of Parallel Programming
0.22	ACM Transactions in Embedded Computing Systems
0.22	Computer Communication Review (ACM SIGCOMM)
0.21	ACM Transactions on Computer Systems (TOCS)
0.21	IEEE Concurrency
0.20	SIGARCH Computer Architecture News

functions in graphs, which could also be used as primitives in context and/or social aware search tasks.

9. 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] L. Backstrom, D. P. Huttenlocher, J. M. Kleinberg, and X. Lan. Group formation in large social networks: membership, growth, and evolution. In *KDD 2006*.
- [5] D. Bader, S. Kintali, K. Madduri, and M. Mihail. Approximating betweenness centrality. In *WAW 2007*.
- [6] Baeza and Ribeiro. *Modern Information Retrieval*. ACM Press / Addison-Wesley, 1999.
- [7] J. Bourgain. On Lipschitz embedding of finite metric spaces in Hilbert space. *Israel Journal of Mathematics*, 52(1):46–52, March 1985.
- [8] U. Brandes. A faster algorithm for betweenness centrality. *Journal of Mathematical Sociology*, 2001.
- [9] V. Chvatal. A greedy heuristic for the set-covering problem. *Mathematics of Operations Research*, 1979.
- [10] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms, 2nd Edition*. The MIT Press, 2001.
- [11] F. Dabek, R. Cox, F. Kaashoek, and R. Morris. Vivaldi: a decentralized network coordinate system. In *SIGCOMM 2004*.
- [12] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1959.
- [13] R. Fagin, A. Lotem, and M. Naor. Optimal aggregation algorithms for middleware. *J. Comput. Syst. Sci.*, 2003.
- [14] R. W. Floyd. Algorithm 97: Shortest path. *Commun. ACM*, 5(6), June 1962.
- [15] D. Fogaras, and B. Racz. Towards Scaling Fully Personalized PageRank. *Algorithms and Models for the Web-Graph*, pp. 105-117, 2004.
- [16] L. Freeman. On the sociological concept of “group”: a empirical test of two models. *American Journal of Sociology*, 98:52–166, 1993.
- [17] L. Freeman. A set of measures of centrality based on betweenness. *Sociometry*, 40(1):35–41, 1977.
- [18] D. Gibson, J. M. Kleinberg, and P. Raghavan. Inferring web communities from link topology. In *HYPERTEXT 1998*.
- [19] M. Girvan and M. Newman. Community structure in social and biological networks. *Proc. Natl. Acad. Sci.*, 99:8271–8276, 2002.
- [20] A. Goldberg. Point-to-Point Shortest Path Algorithms with Preprocessing. *SOFSEM (1)*, 2007.
- [21] 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.
- [22] A. Goyal, F. Bonchi, and L. Lakshmanan. Discovering leaders from community actions. In *CIKM 2008*.
- [23] B. M. Hill. A simple general approach to inference about the tail of a distribution. *Annals of Stat.*, 1975.
- [24] 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.
- [25] W. Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association* 58(301):13–30, 1963.
- [26] J. Hopcroft, O. Khan, B. Kulis, and B. Selman. Natural communities in large linked networks. In *KDD 2003*.
- [27] 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.
- [28] J. Kleinberg, A. Slivkins, and T. Wexler. Triangulation and embedding using small sets of beacons. In *FOCS 2004*.
- [29] R. Kraft, C. C. Chang, F. Maghoul, and R. Kumar. Hierarchical Graph Embedding for Efficient Query Processing in Very Large Traffic Networks. In *SSDBM 2006*.
- [30] HP. Kriegel, P. Kroger, M. Renz, and T. Schmidt. Vivaldi: a decentralized network coordinate system. In *SIGCOMM 2004*.
- [31] R. Kumar, J. Novak, and A. Tomkins. Structure and evolution of online social networks. In *KDD 2006*.
- [32] R. Kumar, P. Raghavan, S. Rajagopalan, and A. Tomkins. Trawling the web for emerging cyber-communities. *Computer Networks*, 1999.
- [33] M. Newman and M. Girvan. Finding and evaluating community structure in networks. *Phys. Rev.*, 69, 2004.
- [34] E. Ng and H. Zhang. Predicting internet network distance with coordiantes-based approaches. In *INFOCOM 2001*.
- [35] M. J. Rattigan, M. Maier, and D. Jensen. Using structure indices for efficient approximation of network properties. In *KDD 2006*.
- [36] H. Samet, J. Sankaranarayanan, and H. Alborzi. Scalable network distance browsing in spatial databases. In *SIGMOD 2008*.
- [37] P. Singla and M. Richardson. Yes, there is a correlation: from social networks to personal behavior on the web. In *WWW 2008*.
- [38] L. Tang and M. Crovella. Virtual landmarks for the internet. In *IMC 2003*.
- [39] M. Thorup and U. Zwick. Approximate distance oracles. In *ACM Symp. on Theory of Computing*, 2001.
- [40] H. Tong, C. Faloutsos and J-Y.Pan. Fast random walk with restart and its applications In *ICDM*, 2006.
- [41] A. Ukkonen, C. Castillo, D. Donato, and A. Gionis. Searching the wikipedia with contextual information. In *CIKM*, 2008.
- [42] J. Venkateswaran, D. Lachwani, T. Kahveci, and C. Jermaine. Reference-based indexing of sequence databases. In *VLDB 2006*.
- [43] 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*.
- [44] S. Wasserman and K. Faust. *Social Network Analysis*. Cambridge University Press, Cambridge, MA, 1994.

- [45] Y. Xiao, W. Wu, J. Pei, W. Wang, and Z. He. Efficiently Indexing Shortest Paths by Exploiting Symmetry in Graphs. In *EDBT 2009*
- [46] U. Zwick. Exact and approximate distances in graphs — a survey. *LNCS*, 2161, 2001.