Adaptive Landmark Selection Strategies for Fast Shortest Path Computation in Large Real-World Graphs

Frank W. Takes and Walter A. Kosters
Leiden Institute of Advanced Computer Science (LIACS), Leiden University, The Netherlands
{ftakes,kosters}@liacs.nl

***Abstract*—This paper considers the task of answering shortest path queries in large real-world graphs such as social networks, communication networks and web graphs. The traditional Breadth First Search (BFS) approach for solving this problem is too time-consuming when networks with millions of nodes and possibly billions of edges are considered. A common technique to address these complexity issues uses a small set of landmark nodes from which the distance to all other nodes is precomputed in order to then answer arbitrary distance queries by navigating via one of the selected landmarks. Although many strategies to select landmarks have been introduced in previous work, the problem of finding an optimal set that covers the entire graph remains NP-hard. Our contribution starts with a study of characteristics that determine the successfulness of a landmark selection strategy. We propose a new adaptive heuristic for selecting landmarks that does not only pick central nodes, but also ensures that these landmarks properly cover different areas of the graph. Experiments on a diverse set of large graphs show that the proposed selection strategy and assisting node processing technique can efficiently estimate the node-to-node distance in graphs with millions of nodes with very high accuracy, while using the same amount of precomputation time as previously proposed strategies.**

***Keywords*-graphs; shortest paths; distances; landmarks**

I. Introduction

A large part of computer science research deals with finding or computing simple paths, shortest paths or distances between objects in a dataset that can be modeled as a graph (also called a network). Applications include computation of graph properties such as the distance distribution or the diameter [1] of the graph. In a web context, an example of a path-related query is a request for a simple yes or no answer to the question of whether or not two nodes are connected, solving the so-called *reachability* problem [2] in graphs, with well-known web-related applications in for example XML parsing and ontology querying [3]. In this paper we will focus on the problem of finding the length of a shortest path between a given pair of nodes, assuming that all nodes in the graph are connected. More specifically, we consider the task of answering *distance queries* in large graphs, which has been shown to be a complex and challenging task [4]. By *large graphs* we mean that the distance matrix cannot be stored in main memory, and algorithms with quadratic space or time complexity in the number of nodes or edges are not acceptable. Typical examples of large graphs include social networks, biological networks, communication networks and web graphs. A typical webgraph can easily have millions of nodes and possibly hundreds of millions or even billions of edges.

The problem of efficiently computing the shortest path length, i.e., the *distance* between two nodes in a graph, has been extensively studied [5]. In unweighted graphs, which we mainly consider in this paper, finding a shortest path originating from a particular node can be done by performing a Breadth First Search (BFS) until the goal node is found. For a graph with n nodes and m edges, one BFS considers each edge at most once, realizing a time complexity of $O(m)$. Doing this for each of the n nodes in the graph results in a complexity of $O(mn)$ for determining the shortest path length between all possible pairs of nodes, essentially solving the well-known All Pairs Shortest Path (APSP) problem. Clearly, for the large graphs that are nowadays studied a traditional brute-force approach is not feasible in terms of time and memory consumption.

Due to these complexity issues, approximation and estimation techniques have been introduced, most notably methods based on so-called *beacons* [6] or *landmarks* [7] that do some precomputation in order to then answer a shortest path query very quickly. Often, a small set of landmarks (nodes) is selected, for which the actual distance to every node is precomputed. When a distance query is received, an approximation based on the distance to and from these landmarks is given, using some smart lower and upper bounds on the landmark distances, assisted by checks for trivial cases in which an exact answer can be returned. Typically, the precomputation step is orders of magnitude faster than computing all the shortest paths, the size of the landmark set is orders of magnitude smaller than the number of nodes n, and the distance query time is orders of magnitude faster than the traditional BFS query time. Ultimately, there is a trade-off between space, precomputation time, query time and accuracy [4].

Although a *random* set of landmark nodes for estimating the shortest path lengths already works quite well [6], it has been shown that a careful selection strategy, for example based on nodes with a high centrality value [7], or based on a tree which covers an as efficient as possible portion of the graph [8], [9] can greatly improve the performance of the landmark method. Clearly, not every selection strategy performs well on every type of graph, and choosing the correct landmark selection strategy can be of great influence on the accuracy of the method for a particular graph. The problem of selecting the *perfect* minimal set of landmarks, has been proven to be NP-hard [7].

Our contribution consists of a careful analysis of what makes a certain landmark selection strategy perform well. We will look at the nodes for which the error is high, and attempt to characterize these nodes by their position in the graph. Based on the obtained insights, we propose two new techniques that attempt to improve landmark selection techniques based on common centrality measures.

The rest of this paper is structured as follows. In Section II we consider some notation and precisely formulate our problem statement and landmark approach. Next, related work is discussed in Section III, after which we explain our landmark framework in detail in Section IV. Most notably, two new landmark strategies are proposed in in Section V. In Section VI we perform a number of experiments to compare the suggested techniques on a set of large real-world graphs. Finally Section VII summarizes the paper and provides suggestions for future work.

II. Preliminaries

In this section, basic notation is briefly discussed, a formal problem statement is given, and finally the landmark approach and its constraints are explained.

A. Notation

We consider an undirected and unweighted graph $G = (V, E)$ with $n = |V|$ nodes and $m = |E|$ edges. Because the graph is undirected, each edge is included twice, so $(u, v) \in E$ iff $(v, u) \in E$. Then, a path is defined as a sequence of nodes connected by edges. A shortest path between two nodes $u, v \in V$ is a path consisting of a minimal number of edges that connects the two nodes. The length of this shortest path, or the *distance* $d(u, v)$ between nodes u and v, is simply the number of edges in such a shortest path. The assumption is that G is connected, i.e., $d(u, v)$ is finite for all nodes u and v. The *degree* $deg(v)$ of a node v is the number of edges connected to that node. We assume that the graphs are sparse, meaning that m is much smaller than the maximum number of edges $n(n-1)$.

B. Problem definition

We consider the problem of accurately and efficiently *estimating* $d(u, v)$ for any given pair of nodes $u, v \in V$. By *accurate*, we mean that the estimated value should not differ too much from the actual distance value, i.e., the error, which we will define more precisely in Section VI-B, has to be as low as possible. By *efficient*, we mean that the computational step of one distance estimation should be significantly faster than one simple BFS, which can be done in $O(m)$. Practically speaking, it should be possible for large graphs to estimate thousands of these distance values in a matter of seconds. The computational step of a distance estimation, which we call the *query time*, should only iterate over the set of nodes (or a subset), meaning that it should be done in at most $O(n)$ time.

To realize a low shortest path computation time, a relatively short *precomputation phase* is allowed. In the precomputation phase, we typically iterate over the set of nodes and/or edges a constant number of times, for example to perform a few real BFS runs (each taking $O(m)$ time). So for a reasonably small integer constant $c > 0$, the *precomputation time* should be restricted to cm. The same requirement holds for the space complexity: the precomputation data should take no more memory than the graph data itself.

C. Landmarks

As a precomputation step, we select a set of landmarks $B \subseteq V$ consisting of $k = |B|$ nodes (with $k \ll n$) for which we precompute for all pairs v, w (with $v \in B$ and $w \in V$) the exact value of $d(v, w)$. Because we deal with undirected graphs, we automatically also compute $d(w, v)$. Note that k is typically very small compared to n, and thus storing $k \times n$ distances is possible. In contrast, storing $n \times n$ distances, i.e., the full distance matrix, is not possible. Indeed, for $k = n$ we would essentially be solving the All Pairs Shortest Path (APSP) problem which is prohibited due to time and memory constraints. The problem of selecting a good set of landmarks B from the original set of nodes V is considered in Section IV and Section V. When we answer a distance query, i.e., a request for finding the distance $d(u, v)$ between two nodes u and v, we first check some trivial cases that can be answered easily (assuming the graph is stored using adjacency lists):

- If u or v is a landmark, then we can return the exact value of $d(u, v)$ as this value is stored for the landmark.
- If u and v are identical, then obviously $d(u, v) = 0$.
- If u and v are direct neighbors (which can be determined in $O(\log m/n)$ by searching for v in the sorted list of neighbors of u (or vice versa), then obviously $d(u, v) = 1$.
- If u and v are at distance 2, then we can also detect this efficiently in $O(m/n)$ as we can iterate over both u and v's sorted lists of neighboring nodes in search for a duplicate entry, resulting in $d(u, v) = 2$.

Any distance larger than 2 will have to be estimated using the landmark set by considering each of the k nodes in the precomputed set of landmarks. As observed in [7], due to the triangle equality, the following statement holds regarding the value of $d(u, v)$ given a set of landmarks B: $\max_{w \in B}(|d(u, w) - d(w, v)|) \leq d(u, v) \leq \min_{w \in B}(d(u, w) + d(w, v))$. Thus, by considering the precomputed distances for the landmarks, we can obtain a lower and upper bound L and U on the distance between u and v. Here, L is at least equal to 3, as otherwise we would have found the distance as one of the trivial cases. So we have:

$$L = \max\big(\max_{w \in B}(|d(u, w) - d(w, v)|), 3\big)$$

$$U = \min_{w \in B}(d(u, w) + d(w, v))$$

When asked for an estimate, we can return L or U itself, the mean, geometric mean, or some other variation using the two variables. It turns out that using U as an estimate gives the lowest error rate [7].

III. Related work

The problem of exactly determining the distance between any or all pairs of nodes has been widely addressed. Initially, algorithms that do fast matrix multiplication [10] were frequently used. Such algorithms improve upon the straightforward Floyd-Warshall algorithm for solving the APSP problem, but suffer from large constants and obvious memory constraints. Other exact approaches are based on A* [11], but still have poor worst-case complexity.

Considering estimation and approximation techniques, data structures for answering distance queries were introduced under the name "distance oracles", providing some theoretical results on the accuracy of the estimation [12]. Although elegant in design, these techniques are not very useful when graphs with many low distance values are considered [6], as the actual difference between the approximated and real distance can be large, which is undesirable in large graphs with relatively low pairwise distances. This happens to be the case in many of the real-world graphs that are nowadays studied, as they usually belong to the class of so-called small world networks [13] with very low average pairwise distances.

Methods based on beacons [6] or landmarks [7], [14] were suggested as a better way of handling distance queries in this type of graphs. Selecting a minimal set of landmarks such that the graph is covered, meaning that the estimate for $d(u, v)$ is correct for all pairs $u, v \in V$, was shown to be NP-hard [7]. Selecting an efficient set of landmarks based on the centrality of a node or based on a "highway" [9] or a tree decomposition [8], [15] of the graph were suggested as heuristics for selecting an optimal set of landmarks. Various optimizations based on pruning the BFS, bitwise tricks and parallelism were introduced in [16]. Furthermore, the landmark selection method in evolving graphs with edge additions and deletions has been described in [17].

The landmark method has clearly been widely addressed, but due to the NP-hardness of the landmark selection problem, it remains a challenging method worth studying.

IV. Landmark framework

This section describes the landmark framework, which consists of two parts: landmark selection and landmark processing. Landmark selection, considered in Section IV-A, deals with the problem of sorting the nodes based on their likeliness of being a good landmark. Section IV-B is about landmark processing, which deals with the question of how and which of the identified landmarks should finally be used. Some minor optimizations are discussed in Section IV-C.

A. Landmark selection

Landmark selection deals with the task of selecting a total of k nodes from the total set of n nodes that are going to serve as landmarks. As an improvement over a *random* selection of k landmarks, several landmark selection strategies based on centrality of the nodes in the graph are suggested in [7]. The idea behind this is to compute the centrality value $C(v)$ of all nodes $v \in V$, and then select the k most central nodes (with the highest value of $C(v)$) as landmarks. In this paper we will consider the following centrality measures:

- Degree centrality: $C_{deg}(v) = deg(v)/(n-1)$.
- Closeness centrality: $C_c(v) = \frac{1}{n-1} \sum_{w \in V} d(v, w)$.
- Betweenness centrality: $C_{bc}(v) = \sum_{u \neq v \neq w} \frac{\sigma_v(u,w)}{\sigma(u,w)}$, where $\sigma(u, w)$ is the number of shortest paths from u to w and $\sigma_v(u, w)$ is the number of shortest paths that run through node v [18].
- PageRank: $C_{pr}(v)$, which is the value of $PR(v)$ after iteratively (usually 100 iterations is enough for convergence) and simultaneously applying $PR(v) \leftarrow \frac{1-d}{n} + d\left(\sum_{w \in N(v)} \frac{PR(w)}{deg(w)}\right)$ for each of the nodes $v \in V$, where $PR(v)$ is initialized to $1/n$ and $N(v)$ is the set of nodes adjacent to node v and the well-known random-surfer parameter d equals 0.15 [19].

Betweenness and closeness are two centrality measures that are just as hard to compute as the distance between all nodes (they require $O(mn)$ time). Luckily both measures can be estimated by means of sampling, reducing complexity to cm where c is the number of samples. Computing the PageRank value of all the nodes also means iterating over the set of m edges a constant number of times, resulting in a similar time complexity.

Although numerous other centrality measures have been suggested in literature, we believe that these four measures are the most common, but more importantly are of four different types. Respectively, they are based on a local property of the nodes (the degree), the number of shortest paths that runs through a node (betweenness), the average distance from the node to every other node (closeness) and the centrality of the node based on a propagation model (PageRank).

Intuitively, the best nodes to be selected as landmarks for finding shortest paths length, would be the nodes with the highest *betweenness centrality* value, as the value of this measure inherently suggests that the node is part of a large portion of all the shortest paths. However, if we consider an error measure which takes the difference between the estimated and real distance into account (see Section VI-B), then nodes on almost-shortest paths (e.g., realizing distance plus one) are also good, but do not necessarily have the highest betweenness value. This suggests that there might be a better landmark selection strategy than simply selecting the nodes with the highest (betweenness) centrality value.

B. Landmark processing

When a set of nodes has been generated based on some (centrality) measure or strategy, a sorted list of nodes can be generated with the most central nodes on top. The simplest form of node selection is then to take the top k nodes (recall that k is the number of landmarks) without any further evaluation of how these nodes are positioned in the graph. In [7], a number of improvements over

this processing technique are suggested. First, one could choose to select as landmarks the highest ranked nodes from the list from each *partition* in the graph as defined by some partitioning or clustering algorithm. Although intuitively useful, this suggested improvement did not produce a significantly much lower error, but comes with an additional computation cost and is thus not considered further in this paper.

A second suggested processing technique is to process the list from top to bottom, but skipping nodes that are at most at distance x from previously selected landmarks. The idea behind this is that a central node that is close to previously selected landmarks does not contribute equally compared to a central node further away from previously selected landmarks. The latter optimization hints towards a second pitfall in simply using the most central nodes as landmarks, namely that central nodes are often direct neighbors. Although it turned out that $x = 1$ performed best, sometimes this processing step gives no improvement or even increases the error.

C. Optimizations

Several optimizations that can be applied after a path based on landmarks has been derived, have been suggested in the literature. Most notably, if for determining $d(u, v)$ the concatenation of paths from node u to landmark w to node v includes the same node more than once, then the intermediary nodes can be skipped, known as *cycle elimination* [14]. Furthermore it is suggested that when two paths have been concatenated, a quick check for a *shortcut* can be done by determining for each node in the path whether its neighborhood contains any of the successors in the path, and if so, using this edge instead of the subpath to that successor.

We introduce an additional seemingly obvious optimization specifically for increasing the bound value for nodes with degree 1. If node u has a degree of 1, then u always needs its direct neighbor to navigate to every other node in the graph. Thus, for this node, the lower or upper bound of the neighboring node plus 1 can be returned. Real-world graphs typically have a power law degree distribution, and nodes with degree 1 are expected to be very common.

V. Balancing Centrality and Covering

The previous section has described two problems when it comes to using the most central nodes as landmarks. First, centrality measures often do not take into account almost-perfect distances and second, most importantly, central nodes are often grouped together, not properly covering different parts of the graph. In this section, improvements for overcoming these two problems are suggested for both the landmark selection and the landmark processing step discussed in the previous section.

A. Adaptive landmark selection

The landmark selection strategy that we propose in this paper is *adaptive*, meaning that the strategy improves its set of landmarks based on the error reduction of its nodes. First, let us look at a preliminary figure of the performance of different selection strategies based on centrality measures in Figure 1. Clearly, the percentage of correctly assessed path lengths (which we will call the *success rate*) increases monotonically with the number of landmarks. An important observation is that centrality measures work regardless of the size of the landmark set: landmarks selected based on centrality measures are able to realize a significantly higher success rate than landmarks that were selected at random. The same was observed for the other real-world graphs that we studied, although there was no single best-performing centrality measure.

We furthermore note that not every landmark appears to evenly contribute to increasing the success rate: some landmarks (steps in Figure 1 from k to $k + 1$ landmarks) contribute significantly more to the success rate than others. Ideally the nodes that realize a big increase in the success rate should be ranked higher than nodes that only marginally increase the success rate. Although obviously the increase realized by a landmark node is highly dependent on previously chosen landmarks, we do expect nodes with a great incremental contribution over previously selected landmarks to give a high contribution to the performance in general. This observation is the basis for the *adaptive landmark selection strategy*:.

1) Sort the set of nodes V based on degree centrality, resulting in a list of ranked nodes, with the most central node v having rank $R(v) = 1$.
2) Perform the sampling phase, in which a number of BFS runs is performed, storing how many times each node v is part of one or more shortest paths.
3) Compute the value of $S(R(v))$, the success rate at each successive rank. Note that here the rank is equal to the potential landmark count. This means that we are generating the plot in Figure 1 for the particular centrality measure chosen in Step 1.
4) For rank $i = 1$ to n, derive for each rank i the value of $\Delta S(R(v))$: the increase in the success rate realized by the landmark node v at rank i.

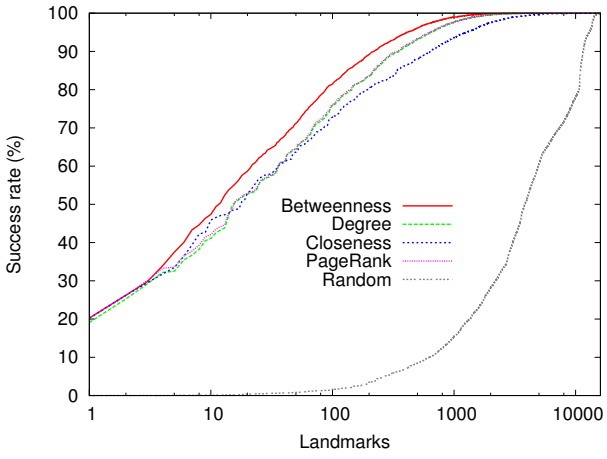

Figure 1. Success rate of different centrality measures as landmark selection strategies, applied to the CA-CondMat network.

5) Re-sort the list of nodes according to $\Delta S(R(v))$, resulting in a list with the (node with the) highest increase in success rate on top.
6) Use the list from step 5 as input for the processing step of the landmark framework, cf. Section IV-B.

In step 1, we have chosen degree centrality, because this measure does not require any additional computation time. Recall that in step 2, counting the number of shortest paths is just as complex as computing the shortest path [18]. Furthermore, performing one BFS from a particular node to every other node, and then comparing the actual distances with the estimated distances, allows us to quickly sample $n-1$ distance computations using only one BFS. Note that the error determination procedure described above replaces the computational step of for example betweenness centrality or closeness centrality. Because the number of samples in step 2 is equal to the number of samples needed to compute the node centrality values of some centrality measure, there is no additional computation time involved in the adaptive landmark selection technique.

The intuition behind this method is that because we first sort the list based on a centrality measure and then compute the error difference of each node in the list, we are taking both the centrality aspect and the covering aspect into account, as nodes that do not significantly reduce the error, apparently do not cover parts of the graph that are not already covered by other nodes. Although step 2 through step 5 could be performed in an iterative process until the error converges to a minimum, we found that one iteration always results in improvement, but more than one iteration almost never improves and sometimes even results in worse performance. The latter is likely due to the fact that after multiple iterations the influence of the centrality measure is lost. Furthermore, an iterative process requires more BFS runs and thus more computation time.

B. Greedy central neighbor processing

Our second contribution is an alternative landmark processing technique called *greedy central neighbor* processing (in short: *gcn*-processing), which works as follows. When processing the list of nodes generated using some strategy or centrality measure, we select for each node in the list h times its most central neighbor, if such a better neighbor exists and if this neighbor is not already a landmark. For example, we select for each node in the node list sorted by the error as a result of the adaptive landmark selection from the previous section, the neighbor with the highest degree (if that degree is higher than the degree of the currently considered node). The intuition behind this method is that this again solves problems with many central nodes being clustered together, not covering the rest of the graph. Note that for each node in the list, one central neighbor at most h hops away is selected, so the landmark count k remains unchanged. Furthermore, the suggested technique is *greedy*: it does not look at the full neighborhood (which would be computationally expensive), but merely repeatedly picks the most central neighbor.

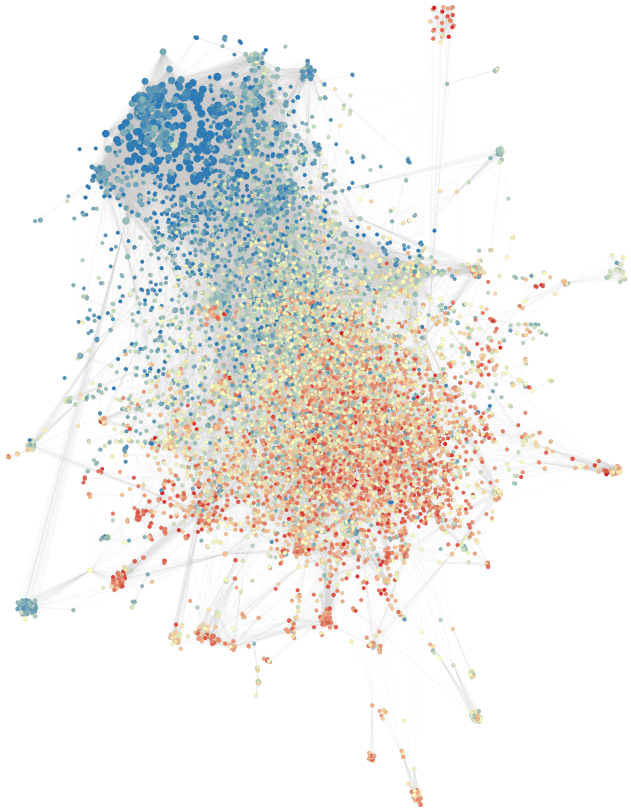

Figure 2. Absolute error in estimated distances in the CA-HEPPH dataset from low (blue) to red (high) with 100 landmarks using *degree centrality*.

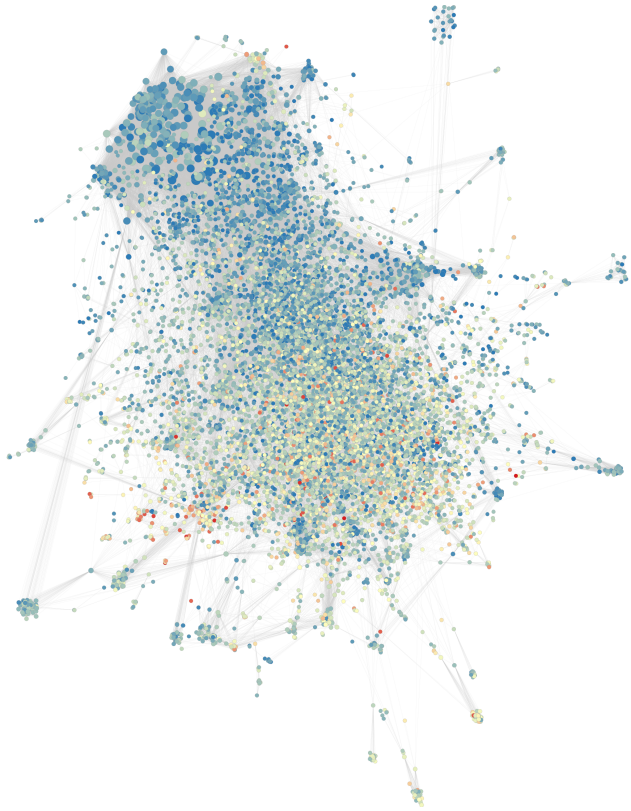

Figure 3. As Figure 2, but with random landmarks and *gcn*-processing. Visualization: ForceAtlas2 in Gephi (`gephi.org`).

As an example, consider Figure 2 and Figure 3 that both show a visualization of the CA-HEPPH network (for a description of this graph, see Section VI-A). In these two visualizations of the same graph, the size of a node is proportional to its degree, and the color of a node denotes the error (see Section VI-B for a description of this error measure) observed when computing a shortest path from or to that particular node. In Figure 2, shortest paths were computed using $k = 100$ landmarks selected using degree centrality, whereas the errors in Figure 3 are from $k = 100$ landmarks that were selected using random node selection, but when processing the list, each time in a greedy way selecting the most central neighbor of the considered node (with $h = 3$). All low error nodes (and thus all high degree nodes) in Figure 2 are grouped together in one highly connected cluster. The error is much higher for the rest of the graph, with higher errors as the distance to the high degree cluster increases. On the other hand, in Figure 3 the error is much lower in the entire graph, demonstrating the usefulness of the greedy central neighbor processing approach in solving the problem of all central nodes residing in one cluster.

In a way, the *gcn*-processing technique combines two centrality measures: one measure is used in the landmark selection phase and another measure is used to guide for greedy central neighbor landmark processing. It is essentially an alternative for taking the weighted average of two measures. In order not to increase the complexity of the precomputation step, the degree can be used as a measure for determining which neighbor has to be selected. In Section VI we will have a detailed look at the performance of this processing step for each of the previously discussed landmark selection strategies.

VI. EXPERIMENTS

In this section we perform experiments on a large set of networks to determine the performance of the different landmark selection techniques, specifically the two new selection and processing techniques introduced in the previous section: adaptive landmark selection and greedy central neighbor processing. We start by describing our datasets in Section VI-A and a verification approach in Section VI-B, after which we discuss the results in Section VI-C.

A. Datasets

Table I gives an overview of the datasets used in this study, including for each dataset a reference to the paper in which the graph's properties are discussed in detail. The second column indicates the type of data represented by the graph. All graphs represent real-world data, are typically very sparse, and adhere to the small world property, meaning that the average distance between any two nodes is very small (less than 8) compared to the number of nodes [13]. For each dataset, of the largest connected component of (the undirected version of) the graph, the number of nodes n, edges m and average node-to-node distance $\overline{d}$ (sampled over $1,000$ node pairs) are listed. We performed experiments for five landmark selection strategies: random selection, betweenness centrality, PageRank, degree centrality, and the newly proposed adaptive landmark selection. For each selection strategy, we experimented with six landmark processing techniques: plain top-k selection (0), skip-1 processing as described in Section IV-B, and greedy central neighbor processing for $h = 2$, $h = 3$, $h = 4$ and $h = 5$, as described in Section V-B. The column "gcn" indicates the lowest error as well as between brackets the value of h for which this error was observed. Closeness centrality never performed better than other measures such as betweenness centrality (see for example Figure 1), so was left out of the result table. Furthermore, random selection with skip-1 processing and degree centrality with *gcn*-processing (which is based again on degree centrality) make no sense, so these respective result columns were also left out.

B. Measurement methodology

In our experiments, we will consistently use 1% of the nodes in the connected component of the graph as landmarks, with a maximum of $k = 100$ landmarks. As this paper specifically considers landmark selection strategies, we do not compare our method with other distance estimation methods. For a general comparison of the landmark framework with such methods, we refer the reader to [7]. Assessing the performance of a landmark strategy can be done by computing the *error* (sampled over $1,000$ node pairs), defined as $|d_{real} - d_{estimate}| \ / \ d_{real}$. Recall that the success rate discussed in Section V-A only counted the number of times a landmark node was on a shortest path. Here, $d_{estimate}$ is the estimated distance (for the real distance d_{real}) by employing the full landmark framework as described in Section IV, so including checks for trivial distances and the described optimizations. Depending on the type of application that is considered, alternative error measures such as counting the percentage of distances that differ by at most 1 could be used.

The number of iterations in the precomputation step is fixed, and the final distance result is measured using the error measure. Therefore, clock time is not a relevant performance measure, nor are the specific properties of the machine used for the experiments. However, to put the results in perspective, we do mention that one BFS using our straightforward C++ code takes about 6 seconds for the 8 million nodes graph in Table I. This means that the total precomputation time for approximating betweenness centrality and the adaptive landmark selection strategy, methods which both perform a total of $k = 100$ BFS runs along with some book-keeping, can be done in a few minutes, even for the largest graph. The same holds for PageRank, which can be computed in roughly the same time, also using 100 iterations. Random landmark selection and degree centrality obviously do not require any precomputation time.

C. Results and discussion

As we already demonstrated for one graph in Section V, random landmark selection is clearly outperformed by

Table I
PERFORMANCE (ERROR, LOWER IS BETTER) OF DIFFERENT LANDMARK SELECTION APPROACHES ON VARIOUS NETWORK DATASETS.

Dataset	Type	n	m	$\overline{d}$	Random		Betweenness			PageRank			Degree		Adaptive		
					0	**gcn** (h)	**0**	**skip-1**	**gcn** (h)	**0**	**skip-1**	**gcn** (h)	**0**	**skip-1**	**0**	**skip-1**	**gcn** (h)
CA-HEPPH [20]	collab.	11.2K	235K	4.66	.509	.080 (3)	**.045**	.078	**.045** (2)	.140	.090	.093 (3)	.137	.091	.117	.103	.080 (3)
GOOGLENW [21]	web	15.7K	297K	2.46	.224	**.000** (3)	.001	.004	**.000** (3)	.001	.003	.001 (2)	.001	.003	**.000**	.003	**.000** (2)
CA-CONDMAT [20]	collab.	21.3K	182K	5.47	.551	.068 (2)	**.044**	.064	.045 (2)	.059	.066	.054 (3)	.100	.098	.064	.083	.056 (3)
CIT-HEPTH [20]	cit.	27.4K	704K	4.29	.562	.040 (4)	.031	.059	**.029** (3)	.048	.069	.046 (3)	.047	.071	.051	.086	.044 (3)
ENRON [22]	comm.	33.7K	362K	4.05	.615	.013 (4)	.010	.009	**.008** (2)	.011	.098	.009 (4)	.012	.102	.022	.145	.012 (3)
SLASHDOT0902 [23]	social	82.2K	1.09M	3.94	.764	.048 (4)	.081	.052	.048 (2)	.085	.046	.053 (2)	.049	.053	.078	.052	**.032** (4)
DBLP [24]	coll.	99.3K	1.09M	3.94	.605	.135 (4)	**.090**	.109	.091 (2)	.103	.105	.099 (3)	.113	.096	.093	.102	.093 (3)
M14B [25]	elec.	100K	1.28M	52.5	1.17	.743 (3)	.276	.116	.267 (3)	.501	.152	.377 (3)	.270	.174	.059	.065	**.054** (2)
WAVE [25]	elec.	156K	2.1M	22.9	.531	.461 (2)	.199	.142	.194 (3)	.270	.126	.211 (3)	.164	.120	.121	**.079**	.096 (3)
WEB-STANFORD [23]	web	255K	3.88M	7.31	.343	.007 (2)	**.003**	.006	**.003** (2)	.005	.007	.006 (2)	.007	.010	.010	.008	.004 (4)
WEB-GOOGLE [23]	web	856K	5.58M	6.18	.884	.149 (3)	.006	.019	**.005** (3)	.006	.006	**.005** (4)	.006	.009	.006	.010	**.005** (4)
WIKI-TALK [23]	comm.	2.39M	9.31M	3.91	.775	**.030** (4)	.038	.122	.038 (3)	.037	.040	.037 (4)	.039	.088	.036	.128	.036 (2)
LIVEJOURNAL [24]	social	4.00M	69.3M	5.39	.831	.163 (3)	**.067**	.079	.067 (2)	.075	.082	.071 (2)	.082	.079	.069	.074	.071 (2)
HYVES [26]	social	8.08M	912M	4.75	.528	.038 (3)	.045	.065	.035 (3)	.035	.055	.035 (2)	.034	.060	.042	.069	**.029** (3)

centrality measures. We note that of the centrality measures, most of the time betweenness centrality has the lowest error. Degree centrality and PageRank are in most cases equally good. Because the newly introduced adaptive landmark selection technique builds upon degree centrality, we say that the new adaptive strategy is successful if it outperforms degree centrality, as otherwise the new method would not be worth the additional computation time compared to degree centrality. As Table I shows, this is the case for all 14 graphs, demonstrating the usefulness of the newly proposed adaptive selection strategy. On a number of datasets, the adaptive landmark selection strategy even outperforms betweenness centrality, a result which strengthens the claim that we made in the beginning of the paper: nodes that are part of a large number of shortest paths do not necessarily serve as the best landmarks. The error observed when using the skip-1 optimization is diverse. Although there is a big increase in performance for the adaptive landmark selection on the WAVE dataset, most of the time the error is similar or worse as was also observed in [7].

The second contribution of this paper is the greedy central neighbor processing strategy. Obviously, this method mainly assists a landmark selection strategy in the final processing phase. To get an idea of the contribution in terms of performance of *gcn*-processing, we can look at the error for a random set of landmarks as compared to a random set of landmarks (the column of Table I titled "Random"). We see that the greedy central neighbor processing technique always greatly improves upon plain random landmark selection, most notably in case of the CA-HEPPH and WIKI-TALK datasets, where the *gcn*-processing technique applied to a random set of nodes even outperforms degree centrality. In all other cases, random selection with *gcn*-processing does not improve upon degree centrality, suggesting that the selected nodes are merely a local minimum. We note that for $h > 4$ there was never an increase in performance compared to smaller values of h.

We also applied *gcn*-processing to the betweenness and PageRank selection methods, and there we also observe increases in performance. Apparently, although betweenness and PageRank both take the global aspect of the graph into account, locally some optimization using the *gcn*-processing technique can still be achieved. In case of the adaptive landmark strategy, the increase in performance obtained by using *gcn*-processing is diverse, but often relevant. For example for the SOC-SLASHDOT or CA-HEPPH network, the error is actually significantly lower when *gcn*-processing is used.

In general we can conclude that the new landmark selection and landmark processing techniques work well for our set of real-world graphs, as shortest path lengths can be determined with an error that is consistently lower than 0.10. We note that even when the average distance in the graph is relatively high, such as for the AMAZON and WAVE datasets, the error remains low. Finally we note that based on the results that we obtained, the error does not appear to be influenced by variables such as the numbers of nodes or edges or the average node-to-node distance, which demonstrates the scalability of the suggested techniques.

VII. CONCLUSION

The performance of the landmark methodology for assessing shortest path lengths in large real-world graphs heavily depends on the chosen landmark selection strategy. Using various experiments we have shown that the task of selecting a good set of landmarks involves at least two aspects: selecting a set of central nodes and properly covering different areas of the graph. In order to address these two aspects, in this paper we have compared different landmark selection strategies and introduced the adaptive landmark selection strategy and the greedy central neighbor processing technique. Experimental results on a number of real-world graphs show that using the same amount of precomputation time, the proposed strategies outperform and improve previously suggested landmark selection techniques based on centrality.

The question remains whether or not it is possible to determine beforehand which landmark strategy is expected to show the best performance. In future work we would like to investigate how we can link different graph-based properties to the performance of a specific selection and processing technique in order to determine a priori a suitable landmark selection strategy. Although we have demonstrated the success of the proposed strategies on a number of real-world graphs, more research is needed to determine the worst-case performance in order to give an upper bound on the error. Furthermore we want to see if our adaptive landmark selection strategy as a whole can also be applied to the problem of determining shortest path lengths in evolving graphs that are growing and shrinking as nodes and edges are added and deleted.

ACKNOWLEDGMENT

This research is part of the COMPASS project, financed by NWO under grant number 612.065.926.

REFERENCES

[1] F. W. Takes and W. A. Kosters, "Determining the diameter of small world networks," in *Proceedings of 20th ACM International Conference on Information and Knowledge Management (CIKM)*, 2011, pp. 1191–1196.

[2] J. Yu and J. Cheng, "Graph reachability queries: A survey," in *Managing and Mining Graph Data*, ser. Advances in Database Systems, 2010, vol. 40, pp. 181–215.

[3] A. Gubichev and T. Neumann, "Path query processing on very large RDF graphs," in *Proceedings of the 14th International Workshop on the Web and Databases (WebDB)*, 6 pages, 2011.

[4] C. Sommer, "Shortest-path queries in static networks," *ACM Computing Surveys*, vol. 46, no. 4, article 45, 2014.

[5] U. Zwick, "Exact and approximate distances in graphs — A survey," in *Proceedings of the 9th European Symposium on Algorithms (ESA)*, 2001, pp. 33–48.

[6] J. Kleinberg, A. Slivkins, and T. Wexler, "Triangulation and embedding using small sets of beacons," in *Proceedings of the 45th IEEE Symposium on Foundations of Computer Science (FOCS)*, 2004, pp. 444–453.

[7] M. Potamias, F. Bonchi, C. Castillo, and A. Gionis, "Fast shortest path distance estimation in large networks," in *Proceedings of the 18th ACM International Conference on Information and Knowledge Management (CIKM)*, 2009, pp. 867–876.

[8] T. Akiba, C. Sommer, and K. Kawarabayashi, "Shortest-path queries for complex networks: Exploiting low tree-width outside the core," in *Proceedings of the 15th International Conference on Extending Database Technology (EDBT)*, 2012, pp. 144–155.

[9] R. Jin, N. Ruan, Y. Xiang, and V. Lee, "A highway-centric labeling approach for answering distance queries on large sparse graphs," in *Proceedings of the ACM International Conference on Management of Data (SIGMOD)*, 2012, pp. 445–456.

[10] D. Aingworth, C. Chekuri, P. Indyk, and R. Motwani, "Fast estimation of diameter and shortest paths (without matrix multiplication)," *SIAM Journal on Computing*, vol. 28, no. 4, pp. 1167–1181, 1999.

[11] A. V. Goldberg, H. Kaplan, and R. F. Werneck, "Reach for A*: Efficient point-to-point shortest path algorithms," in *Proceedings of the SIAM Workshop on Algorithm Engineering and Experiments (ALENEX)*, 2006, pp. 129–143.

[12] M. Thorup and U. Zwick, "Approximate distance oracles," *Journal of the ACM*, vol. 52, no. 1, pp. 1–24, 2005.

[13] J. Kleinberg, "The small-world phenomenon: An algorithm perspective," in *Proceedings of the 32nd ACM Symposium on Theory of Computing (STOC)*, 2000, pp. 163–170.

[14] A. Gubichev, S. Bedathur, S. Seufert, and G. Weikum, "Fast and accurate estimation of shortest paths in large graphs," in *Proceedings of the 19th ACM International Conference on Information and Knowledge Management (CIKM)*, 2010, pp. 499–508.

[15] L. Fu and J. Deng, "Graph calculus: Scalable shortest path analytics for large social graphs through core not," in *Proceedings of the IEEE International Conference on Web Intelligence (WI)*, 2013, pp. 417–424.

[16] T. Akiba, Y. Iwata, and Y. Yoshida, "Fast exact shortest-path distance queries on large networks by pruned landmark labeling," in *Proceedings of the ACM International Conference on Management of Data (SIGMOD)*, 2013, pp. 349–360.

[17] K. Tretyakov, A. Armas-Cervantes, L. García-Bañuelos, J. Vilo, and M. Dumas, "Fast fully dynamic landmark-based estimation of shortest path distances in very large graphs," in *Proceedings of the 20th ACM International Conference on Information and Knowledge Management (CIKM)*, 2011, pp. 1785–1794.

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

[19] L. Page, S. Brin, R. Motwani, and T. Winograd, "The PageRank citation ranking: Bringing order to the web," *Technical Report, Stanford Digital Library Technologies Project*, 1999.

[20] J. Leskovec, J. Kleinberg, and C. Faloutsos, "Graphs over time: Densification laws, shrinking diameters and possible explanations," in *Proceedings of the 11th ACM International Conference on Knowledge Discovery in Data Mining (KDD)*, 2005, pp. 177–187.

[21] G. Palla, I. J. Farkas, P. Pollner, I. Derenyi, and T. Vicsek, "Directed network modules," *New Journal of Physics*, vol. 9, no. 6, p. 186, 2007.

[22] B. Klimt and Y. Yang, "The Enron corpus: A new dataset for email classification research," in *Proceedings of the 15th European Conference on Machine Learnining (ECML)*, 2004, pp. 217–226.

[23] J. Leskovec, K. Lang, A. Dasgupta, and M. Mahoney, "Community structure in large networks: Natural cluster sizes and the absence of large well-defined clusters," *Internet Mathematics*, vol. 6, pp. 29–123, 2009.

[24] J. Yang and J. Leskovec, "Defining and evaluating network communities based on ground-truth," in *Proceedings of the KDD Workshop on Mining Data Semantics*, article 3, 2012.

[25] C. Walshaw, "The graph partitioning archive," `http://staffweb.cms.gre.ac.uk/~c.walshaw/partition` [accessed February 28, 2014].

[26] F. W. Takes and W. A. Kosters, "Identifying prominent actors in online social networks using biased random walks," in *Proceedings of the 23rd Benelux Conference on Artificial Intelligence (BNAIC)*, 2011, pp. 215–222.