Finding Top-k Shortest Path Distance Changes in an Evolutionary Network

Manish Gupta[1]*, Charu C. Aggarwal[2], and Jiawei Han[1]

[1] University of Illinois at Urbana-Champaign, IL, USA
gupta58@illinois.edu, hanj@cs.uiuc.edu
[2] IBM T. J. Watson Research Center, NY, USA
charu@us.ibm.com

Abstract. Networks can be represented as evolutionary graphs in a variety of spatio-temporal applications. Changes in the nodes and edges over time may also result in corresponding changes in structural garph properties such as shortest path distances. In this paper, we study the problem of detecting the top-k most significant shortest-path distance changes between two snapshots of an evolving graph. While the problem is solvable with two applications of the all-pairs shortest path algorithm, such a solution would be extremely slow and impractical for very large graphs. This is because when a graph may contain millions of nodes, even the storage of distances between all node pairs can become inefficient in practice. Therefore, it is desirable to design algorithms which can directly determine the significant changes in shortest path distances, without materializing the distances in individual snapshots. We present algorithms that are *up to two orders of magnitude* faster than such a solution, while retaining comparable accuracy.

1 Introduction

The problem of network evolution [1, 7, 12, 17] has seen increasing interest in recent years of the dynamic nature of many web-based, social and information networks which continuously change over time. The evolution of such networks may also result in changes in important structural properties such as pairwise shortest-path distances. In this paper, we will study the problem of finding the top k shortest path distance changes in an evolutionary network. This problem may be interesting in the context of a number of practical scenarios:

– Social and information networks are inherently dynamic, and the change in shortest paths between nodes is critical in understanding the changes in connections between different entities. It can also be helpful for tasks such as dynamic link prediction modeling with the use of shortest-path recommendation models or in providing insights about new events in the social graph. For example, the distance between tags in an inter-tag correlation graph may change because of tagging events, or the distance between actors in IMDB[3] may change because of the introduction of new movies. Similarly, change in distances on word co-occurrence graphs for microblogs (such as Twitter[4] tweets) can result in better detection and summarization of events.

* Work was partially done during employment at IBM T. J. Watson Research Center
[3] http://www.imdb.com/
[4] http://twitter.com

– Many developing countries have witnessed a rapid expansion in their road networks. An example would be the well-known *Golden Quadrilateral (GQ)*[5] project in India. The detection of important distance changes may provide further insights about connectivity implications. For example, given multiple plans to set up a road network, one can measure the utility of a proposed plan in a particular marketing scenario.
– The launch of collaborative programs changes the structure of virtual collaboration networks. The underlying changes provide an understanding of critical connections between collaborative entities (e.g., authors in *DBLP*) and their evolution.

The detection of interesting hot-spots for which average distance to other parts of the network has changed suddenly is an interesting problem. This is closely related to the problem of finding the top-k maximum distance change node pairs. We will present a number of algorithms for this problem in this paper. We compare our algorithms on a number of real data sets.

We note that the problem of shortest path change can be solved directly by running well known all-pair shortest path algorithms and simply comparing the distance changes between all pairs. However, this is not a practical solution for very large graphs. For example, for a graph containing 10^8 nodes, the number of possible node pairs would be 10^{16}. The complexity of the all-pairs shortest path computation increases at least quadratically [2] with the number of nodes. Furthermore, the storage of such pairwise paths can be impractical. While this has not been a problem with the small memory-resident graphs which are frequently used with conventional algorithms, it is much more challenging for the large-scale networks which arise in social and information network scenarios. In fact, in our experiments, we found it virtually impossible to use such brute-force algorithms in any meaningful way. Therefore, the goal of the paper is to enable practical use of such algorithms in large-scale applications.

The remainder of this paper is organized as follows. In Section 2, we provide an overview of the related work. We introduce our basic algorithm, the Incidence Algorithm and a randomized algorithm to estimate importance of an edge in a graph in Section 3. In Section 4, we discuss various algorithms for ranking of nodes which can potentially be a part of the top-k node pairs. We present our experiments on large graphs in Section 5 and finally conclude with a summary of the work in Section 6.

2 Related Work

The problem of finding the top-k node pairs with maximum shortest path distance change can be solved by a straightforward applications of two instances of the all-pairs shortest path (APSP) problem. Clearly, the running time is sensitive to the method used for APSP computation. Consider a graph containing n nodes and m edges. One can use a variety of methods such as Shimbel's algorithm [20], Dijkstra's algorithm [11], Johnson's algorithm [15], or the Floyd and Warshall [14, 21] algorithms, all which require at least $O(n \cdot m)$ time. Such running times are not very practical for very large graphs containing millions of nodes. A randomized algorithm by Cohen [10] allows us to compute the number of nodes at a distance d from each of the nodes in the graph. While such an approach can be used to approximately determine a superset of the relevant node pairs, a part of the method requires $O(mlog(n) + nlog^2(n))$ time. This is quite inefficient.

[5] http://en.wikipedia.org/wiki/Golden_Quadrilateral

Our problem is also related to that of finding time-dependent shortest paths [25, 26] in a dynamic graph. A number of algorithms in computer networks [22, 8] also solve the problem of recomputing shortest path trees when edges are added to or removed from the graph. Some related work [6, 3, 19, 4] for this problem proposes methods for exact computation of dynamic shortest paths in a variety of graph settings and in parallel or distributed scenarios. Our problem is however that of finding the maximum shortest path *change* between pairs of nodes, rather than that of designing incremental algorithms for *maintaining* shortest paths.

The problem of shortest path distance change is also related to that of outlier detection, since unusually large changes in distances can be considered abnormal behavior. Some work has also been done on node outlier detection in *static* weighted networks [5]. Anomalies which are caused by *dynamic behavior* such as label modifications, vertex/edge insertions and vertex/edge deletions have been studied in [13]. In [18], the authors detect anomalies such as missing connected subgraphs, missing random vertices and random topological changes over web crawl snapshots by measuring the amount and the significance of changes in consecutive web graphs.

As we will see later, we need to design methods for measuring the betweenness of edges in order to determine key changes in the network. A number of betweenness measures have been proposed in [23, 24], though the methods for computing such measures are too slow to be of practical use in very large scale applications.

3 Shortest Path Evolution: Model and Algorithms

Before discussing the problem further, we will introduce some formal notations and definitions. Consider an undirected connected graph G with snapshots $G_1(V_1, E_1)$ at time t_1 and $G_2(V_2, E_2)$ at time t_2. For the purpose of this paper, we only consider the case where new nodes and edges are added to the graph, and they do not get removed. This is quite often the case in many natural information networks such as *IMDB* movie network or *DBLP co-authorship* network in which objects are constantly added over time. Each edge e can be expressed as a three-tuple (u, v, w) where u and v are the nodes on which the edge is incident and w is the weight of the edge. The edge weights denote the distance between two nodes and can only decrease over time. Let $d_1(u, v)$ and $d_2(u, v)$ denote the shortest path distances between nodes u and v in snapshots G_1 and G_2. Let the distance change between nodes u and v be denoted by $\Delta d(u, v)$. We aim to find these top-k node pairs (u, v) with the largest $\Delta d(u, v)$, so that there exists no pair (u', v') where $u \neq u'$ and/or $v \neq v'$, s.t. $\Delta d(u', v') > \Delta d(u, v)$. [6]

One of the keys to determining the node pairs with the most change is to determine the *critical edges* which lie on the shortest paths between many pairs of nodes. Clearly the addition of such edges can lead to tremendous changes in the shortest path distances. Therefore, we define the concept of *edge importance* as the probability that an edge will belong to some shortest path tree. In this section, we first propose a randomized algorithm for edge importance estimation. Then, we will leverage this notion to propose the Incidence Algorithm. This is followed by a method to improve its accuracy. Finally, we propose node ranking algorithms to improve the efficiency of the Incidence Algorithm.

[6] Suppose that $\Delta(u, v)$ is large, as a result of addition of a new set S of edges. It may superficially seem that pairs of the form (node near u, node near v) would also have huge distance changes (since they are related in a somewhat similar way to S) and may swamp the top-k. However, when we consider the effect of multiple new edges in a dense real graph, the swamping effect is not quite as explicit. This suggests that subtle structural effects play an important role in defining the solutions of the underlying problem.

3.1 Edge Importance Estimation Algorithm

We first define some concepts which relate to edge-importance.

Definition 1 (Importance number of an edge). *The Importance Number of an edge $e \in E$ in a graph $G(V, E)$ is the probability that the edge will lie on a randomly chosen shortest path tree in the graph.*

Let us denote the shortest path tree rooted at any node x as the source node by SPT_x. The *Edge importance* $I(e)$ for an edge e can be accurately estimated by using the edge importance measured over a random sample of shortest path trees for the same graph. We will present detailed results in Section 5 which show that a random sampling approach works quite well. For estimating the edge importance numbers, we first choose a set of α nodes randomly into a set S and initialize the importance numbers for all edges to 0. For each of the nodes x in set S, we run the well known Dijkstra algorithm over the graph G by using x as the source node.

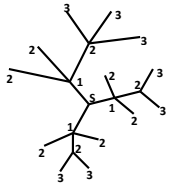


Fig. 1. Distance labeling on SPT_S for a unit weighted undigraph

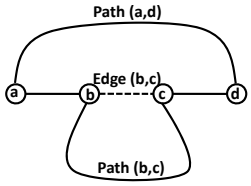


Fig. 2. Small example

Algorithm 1 Edge importance number estimation

1: Input: Graph $G(V, E)$ with n nodes and m edges.
2: Randomly sample α nodes from the graph into set S.
3: Initialize importance number $I(e)$ of every edge $e \in E$ as 0.
4: **for** each node $x \in S$ **do**
5: Run Dijkstra with x as the source node in $O(m + n)$ time to get a shortest path tree SPT_x
6: Label the nodes of SPT_x with the distance from the source node x. Figure 1 shows the labeling for a shortest path tree on a unit weighted undirected graph.
7: For each edge $(i, j) \in E$, identify if (i, j) is an alternative tight edges for SPT_x. Tight edges can be discovered in $O(m)$ time.
8: Choose β random shortest path trees by perturbing the tree SPT_x by replacing γ number of edges in SPT_x each by one of the alternative tight edges.
9: **for** every alternative shortest path tree SPT_{xy} chosen in Step 8 **do**
10: Estimate importance number $I(e, SPT_{xy})$ for every edge e wrt SPT_{xy} as the number of its descendants normalized by n. This takes $O(m)$ time.
11: Add $I(e, SPT_{xy})$ to $I(e)$.
12: **end for**
13: **end for**
14: Compute avg. importance numbers for each edge by normalizing $I(e)$ by $\alpha\beta$.
15: **return** Average importance for each edge $e \in E$

Definition 2 (Distance Label). *The distance label $d^x(i)$ for a node i in a shortest path tree SPT_x is defined as the shortest path distance of the node from the source node x. Note that distance labels are valid if $d^x(j) \leq d^x(i)+w(i, j)$ or $d^x(i) \leq d^x(j)+w(i, j)$ where $d^x(i)$ and $d^x(j)$ are the distance labels for nodes i and j in SPT_x.*

Definition 3 (Tight edge). *An edge (i, j) is a tight edge with respect to shortest path tree SPT_x, if $d^x(j) = d^x(i) + w(i, j)$ or $d^x(i) = d^x(j) + w(i, j)$.*

Definition 4 (Alternative tight edge). *An edge (i, j) is an alternative tight edge if it is a tight edge, but it is not a part of the current shortest path tree.*

After constructing the shortest path trees, we perform distance labeling on SPT_x (as shown in Figure 1). Further, we can generate alternative shortest path trees for SPT_x by replacing tree edges with alternative tight edges. Step 7 in Algorithm 1 iterates over the entire edge set of the graph and checks if it is an alternative tight edge. Step 8 generates random shortest path trees by replacing one or more edges in SPT_x by one of the alternative tight edges.

Observation 1 *For every alternative tight edge, there is a corresponding edge in SPT_x, which can be replaced to create a new perturbed shortest path tree.*

We note that the edge to be replaced can be determined by adding the alternative tight edges, and removing one of the tree edges from the cycle thus created.

We generate a random perturbation of original SPT_x by selecting alternative tight edges for some randomly selected edges of SPT_x. This can be achieved easily by considering the subgraph of all tight edges, and picking one which forms a shortest path tree. We consider β such shortest path trees for each SPT_x.

Definition 5 (Descendant of an edge). *A descendant in the context of an edge (i, j) and a shortest path tree SPT_x is defined as any node that lies in the subtree rooted at the end-point of the edge farther from the source node x.*

The concept of edge descendant is useful, because it measures the importance of the edge (i, j) in connecting nodes contained in SPT_x. Specifically, the number of descendants of edge (i, j) provides a clear understanding of its connectivity level from the source node x to other nodes of the tree. Step 10 computes edge importance as the normalized number of descendants. Let $I(e, SPT_{xy})$ denote the importance of an edge for the y^{th} SPT which is obtained by a random perturbation of SPT_x. This is computed as the ratio of number of descendants for edge (i, j) with respect to SPT_{xy} to $|V|$. $0 \leq I(e, SPT_{xy}) \leq 1$. Finally, in Step 14, average importance numbers are computed for every edge by computing average across all the sampled shortest path trees. Thus, the average importance of an edge is estimated as $I(e) = \frac{\sum_x \sum_y I(e, SPT_{xy})}{\alpha\beta}$. The Importance estimation algorithm runs in $O(\alpha\beta m)$ time. We will use these importance numbers later to improve the Incidence Algorithm described in the next subsection and also for node ranking algorithms.

3.2 The Incidence Algorithm

In this subsection, we present the Incidence Algorithm (Algorithm 2).

Definition 6 (Active node). *A node is **active** if new edges or edges with changed weights are incident on it.*

The intuition for the *Incidence Algorithm* is that the maximum distance change node pairs will include at least one node as an active node with high probability.

Observation 2 *Node pairs containing at least one active node can cover most of the top-k node pairs with maximum distance change with high probability for small values of k. This is particularly true for more dense networks.*

Example: Consider a path $a - b - c - d$ as shown in Figure 2. The solid lines show the shortest paths between the nodes in snapshot G_1. (a, b) and (c, d) are the edges in graph G_1 while (b, c) appears as a new edge in graph G_2. Let $Path(a, d)$ and $Path(b, c)$ be the shortest paths between the corresponding nodes in graph G_1.

Let us assume that the new shortest path between nodes b and c as well as the one between nodes a and d passes through the edge (b, c). Then $d_1(b, c)$ should follow the inequality: $d_1(a, d) - w(a, b) - w(c, d) \leq d_1(b, c) \leq d_1(a, d) + w(a, b) + w(c, d)$. Note that $\Delta(a, d) = d_1(a, d) - w(a, b) - w(c, d) - w(b, c)$ while for $\Delta(b, c)$, we have $d_1(a, d) - w(a, b) - w(c, d) - w(b, c) \leq \Delta(b, c) \leq d_1(a, d) + w(a, b) + w(c, d) - w(b, c)$. Thus, we observe that $\Delta(b, c) \geq \Delta(a, d)$. The distance change would be equal only if shortest path between a and d contained the shortest path between nodes b and c in graph G_1. The changed shortest distance paths in graph G_2 would make use of one or more of the new edges. Using the above example, we can show that the distance change between the endpoints of any new edge would always be greater than or equal to distance change between any other node pair. As shown in Section 5 this is generally true in a wide variety of scenarios, though it may not be true across all network structures.

Algorithm 2 Incidence Algorithm

```
Input: Graphs G1(V1, E1) and G2(V2, E2), Active node set V'.
HEAP h ← φ
for every node n ∈ V' do
    Run Dijkstra Algorithm from n on G1 and G2.
    for every node v ∈ V1 ∩ V2 do
        h.insert((n, v), Δ(n, v)) (Regularly clean heap to control size)
    end for
end for
return top-k pairs (u, v) with maximum Δ(u, v)
```

While the *Incidence Algorithm* provides a decent first-approximation, it is rather naive in its approach. To improve the accuracy, we can consider active node set as the seed set and try to expand from this seed set to include neighbors of nodes in active node set. This expanded seed set is used for determining the source nodes for the different runs of the Dijkstra shortest path algorithm. The goal is to consider only promising neighbors of promising nodes from the active node set for expansion. We will discuss below how this selective expansion is performed.

3.3 Selective Expansion of Active Node Set V'

Without any expansion from the active node set, we just detect the epicenters of shortest path distance changes. However, we may wish to find out node pairs that surround these epicenters too. We use the following method for selective expansion. We first run Dijkstra algorithm from each of the currently active nodes and compute the current top-k shortest path distance change node pairs. Within these top-k, we look at the active nodes and can expand from them. However, we would like to expand one node at a time. Hence, we need to rank the neighbors of the currently active nodes. We would select the neighbor with the highest rank as the source node for the next run of the Dijkstra algorithm. The rank of a neighbor node (say a in Figure 2 which is a neighbor of node b where $a \notin V'$ and $b \in V'$) should depend on the probability that a large number of shortest paths from this node would use the edge (a, b).

Thus, the rank of a neighbor a would be computed as $rank(a) = \frac{I(edge(a,b))}{\sum_{x \in nbr(a)} I(edge(a,x))}$. Then we simply choose the node with the maximum rank and use it as the source node

for running Dijkstra algorithm. We update the top-k node pairs using the new node pairs obtained from the latest Dijkstra run. Node a also becomes active. If the top-k node pair list changes, we choose a new node again, else we terminate the selective expansion.

4 Node Ranking for Improved Efficiency

The *Incidence Algorithm* with selective expansion helps us to obtain the top-k node pairs with high accuracy. However, it is still not quite efficient. If the snapshots of the graph are taken after long intervals, new edges would be incident on a large percentage of the nodes in the graph. The efficiency of our *Incidence Algorithm* is dependent upon the size of this set. Hence, when solving the problem over snapshots of a graph taken over longer time intervals, the *Incidence Algorithm* would be computationally expensive. In this section, we discuss strategies to rank the nodes in the active node set so that the order helps to select the top few nodes which can be used to run single-source based shortest path algorithms and capture the maximum distance change pairs. The trade-off is between the number of nodes selected and accuracy. The goal is to rank the nodes, so that by processing them one by one we obtain more accurate results by running shortest path algorithms from a very small number of source nodes. In the following, we discuss some of the ranking strategies that we used in order to achieve this goal.

4.1 Edge Weight Based Ranking (EWBR)

A node has a higher probability of contributing to the top-k node pairs with maximum shortest distance path change if a large number of *new* low-weight edges are incident on this node. The higher the number of new edges incident on the node, the greater the likelihood that the distances of the other nodes to this node have changed. Of course, such an approach is quite simple, and may have its own drawbacks. For example, an edge contributes only a small part to the shortest path length. So, for graphs with longer shortest path distances, a path would typically consist of a large number of edges and hence the greedy approach of just considering the first edge in that path may not be sufficient. This leads to low node ranking accuracy.

4.2 Edge Weight Change Based Ranking (EWCBR)

A node with a large number of edges whose weight has changed by relatively larger amounts is more important. If the edge weight decreases by a large amount, this edge would naturally become a part of more number of shortest paths. We note that the weight of an edge corresponds to the distance along it. Of course, this distance could be defined differently for different applications. For example, for a co-authorship application, the distance could be the inverse of the collaboration frequency. For edges where one of the nodes was not present in the previous snapshot, change in similarity (1/weight) is set to similarity in the new snapshot. This essentially implies that the similarity in the old snapshot is considered as 0.

4.3 Importance Number Based Ranking (INBR)

The previous algorithm does not distinguish between the different kinds of edges. As discussed earlier, the edge importance is estimated as its likelihood of occurring along

a shortest path. A node has a higher probability of contributing to the top-k node pairs with maximum shortest distance path change, if a large number of new (or weight-changed) important edges are incident on this node. The importance is measured on the new snapshot. Thus, ranking nodes in this order and considering the top few nodes would ensure that we are capturing the effect of most of the important edges. If importance numbers follow a power law, then contributions to distance changes by the tail of this edge ordering should be minimal. Therefore, the consideration of only a top few should provide us high accuracy.

4.4 Importance Number Change Based Ranking (INCBR)

If an edge has a lower importance number in the old graph snapshot and now its importance number has increased a lot, it implies that the edge is important with respect to our task. Similarly, edges with high edge importance scores in old snapshot and low scores for the new snapshot are also interesting. Active nodes with large number of such new or weight-changed edges become important. Note that the importance numbers in the old snapshots for edges that are completely new (i.e., not just weight changes) is considered to be 0.

4.5 Ranking Using Edge Weight and Importance Numbers (RUEWIN)

To rank a node, we can use both the number of important edges and the weight of the new or weight-changed edges incident on a node. Apart from the absolute values of the two quantities, we can also use the actual change in the quantities for node ranking. RUEWIN uses change in weight multiplied by absolute values of importance numbers in the new snapshot for ranking, while RUEWINC uses change in edge weight multiplied by change in importance numbers for ranking.

4.6 Clustering Based Ranking (CBR)

Intuitively a new inter-cluster edge with low weight would be more important in reducing the shortest path distance between a pair of nodes compared to an intra-cluster edge. This is because nodes within a cluter are already well connected, and nodes across clusters have high likelihood to be used in a shortest path. In other words, an edge which connects two regions of high density is more important than an edge which connects nodes within a high density region.

In this scheme of ranking, we first partition the graph using a minimum cut based algorithm. We use METIS [16] for this purpose. From each of the partitions, we randomly choose one of the nodes (called the representative node of the cluster) on which at least one new edge is incident. If we do not perform partitioning, we can randomly select initial nodes from the entire set of nodes on which new edges are incident. We call the approach with partitioning as CBRP and the one without partitioning as CBR. The Dijkstra algorithm is run with the representative node as the source node. Other nodes on which new edges are incident are assigned to clusters whose representative nodes are the closest to the current node.

Now, we need to estimate the change in distance that occurs between two nodes because of a new edge and use this measure as the importance of the new edge. Inter-cluster distance is computed as the distance between the representative nodes of the two

clusters. Distance change is computed as the estimated old distance between the pair of nodes on which the new edge is incident minus the weight of the new edge. The old distance between the two nodes corresponding to an inter-cluster edge is estimated as the sum of the distances of the nodes from the representative nodes of their respective clusters and the inter-cluster distance between the corresponding clusters. We compute the old distance corresponding to an intra-cluster edge in the same way, except that both the nodes belong to the same cluster. Finally, the cluster based score of a node is determined as the sum of the importance of the new edges incident on the node. Note that in this method, the estimation of the old distance for an intra-cluster edge can be very inaccurate. The accuracy of the estimate depends on the size of the cluster. If the cluster has a high radius, the estimate would be bad. However, the intra-cluster edges are not really important for finding the top-k node pairs for which maximum shortest path distance change has occurred, unless they belong to clusters with very large radius. We experimented and noticed that relative estimates of shortest path distances are quite similar to actual distances. Also, we observed that the radius of the clusters are comparatively smaller when we use partitioning based method to choose the initial set of source nodes rather than making a random choice from all over the graph. Lower radius of clusters means better estimates of shortest path distances between nodes in the old graph snapshot, which leads to better accuracy.

The preprocessing steps involved in this form of ranking are: (1) A graph partitioning of the old snapshot using METIS (2) c Dijkstra algorithm runs where c is the number of clusters. Also note that, for an edge that connects an old node to a new node, we cannot use the above logic to estimate the distance saved. Hence, we set their cluster based scores to 0.

5 Experimental Results

The goal of the section is to show the practical usability and efficiency of the method in large scale systems at the expense of a modest loss of accuracy. In order to provide further insights, we also present several intermediate results which show that in many real data sets, a recurring theme is that there are often only few node pairs which share the bulk of the changes in shortest path distances. This suggests that our ranking approach for finding these nodes is likely to be efficient and accurate.

5.1 Datasets

We used the *DBLP co-authorship graphs*[7], *IMDB co-starring graphs*[8] and *Ontario road network graphs*[9]. The nodes for the *DBLP co-authorship graphs* are authors and the edges denote the collaborations. The edge weight is the reciprocal of the co-authorship frequency. We used five pairs of the co-authorship graphs from 1980 to 2000 for testing purposes. The nodes for the *IMDB co-starring graphs* are actors and actresses and edges denote the collaborations. The edge weight is the reciprocal of the co-starring frequency. We consider the co-starring graphs from 1950 to 1952. The nodes for the *Ontario Road Network* are road intersections and edges denote the road segments. The Geography

[7] http://www.informatik.uni-trier.de/∼ley/db/
[8] http://www.imdb.com/
[9] http://geodepot.statcan.gc.ca/

Markup Language (GML) files for this road network for 2005 and 2008 were obtained from the Canada Statistics website[10]. The dataset are defined in terms of latitudes and longitudes, which were converted into edge lengths for the purpose of the algorithm. Edges from 2005, which were missing from the 2008 snapshot, were added to the data set.

We provide details of the characteristics of these graphs in Tables 1, 2 and 3. In the table, the set of edges with change in weight is denoted by E_{1C}(**c=change**). The set of edges which are absent in G_1 and for which one of the nodes was in G_1 is denoted by E_{2ON}(**o=old, n=new**). The set of edges which were absent in G_1 but both the nodes were present in G_1 is denoted by E_{2OO}.

Table 1. *DBLP* Details

Year	Nodes in LCC	Edges	E_{1C}	E_{2ON}	E_{2OO}	$\|V'\|$	Max freq
1980	4295	8096					45
1981	5288	10217	607	953	305	895	46
1984	10598	21592					52
1985	13025	27260	1272	2271	616	2128	56
1988	22834	50255					61
1989	27813	61704	3282	5016	1921	4963	63
1993	61592	148287					87
1994	74794	183738	10995	15446	6569	14633	111
1999	163203	456954					221
2000	188048	539957	32876	39542	21004	35723	231

Table 2. *IMDB* Details

Year	Nodes in LCC	Edges	E_{1C}	E_{2ON}	E_{2OO}	$\|V'\|$	Max freq
1950	144991	8.2M					739
1951	149260	8.6M	111097	74757	179698	16364	745
1952	154719	9.1M	118276	146661	207859	17596	745

Table 3. *Ontario RN* Details

Year	Nodes in LCC	Edges	E_{1C}	E_{2ON}	E_{2OO}	$\|V'\|$	Max freq
2005	348236	455804					250
2008	367628	494067	2280	11870	9041	29539	250

We note that the *DBLP* and *IMDB* graphs are naturally temporal, and therefore a *cumulative graph* can be defined which aggregates all edges upto time t in order to create a graph. The change is then computed between two such cumulative graphs. For example, a *DBLP* graph at year 1980 captures all the co-authorship relationships in *DBLP* until the year 1980. In the case of the *Ontario road network*, the two snapshots in 2005 and 2008 were used in order to determine the change.

The ground truth (also referred to as the "golden set") in terms of the node-pairs with maximum change in distance value was determined. We used the fast implementation mentioned in [9] to generate this golden set. As mentioned earlier, this is the alternative (but brute-force method) for determine the precise node pairs with the maximum distance changes. It is important to note that *this process required several days on multiple CPUs to compute the exact distance changes using direct applications of shortest path algorithms.* Our computational challenges in determining the ground truth is itself evidence of the difficulty of the problem.

5.2 Evaluation Methodology

We compute the accuracy of the results by computing the top-k pairs with the greatest change in the original data (also known as the ground truth or golden set), and compar-

[10] http://tinyurl.com/yfsoouu

ing it with the results of our algorithm. We refer to the fraction of such matching node pairs as the top$K$Accuracy. We will show the tradeoffs between the running time and the top$K$Accuracy achieved by the method. Since the core of our algorithm is to rank and select (the top-ranking) nodes for use as source nodes, one proxy for the time spent is the number of nodes from which the shortest path algorithm needs to be executed. Hence, for every ranking method, we plot the top$K$Accuracy against the number of source nodes. The *area under such a curve* is also a proxy for the average lift achieved with the use of a small number of source nodes. Thus, we compare various ranking algorithms with respect to these areas. Before, describing these results in detail, we will provide some intermediate results which provide some interesting insights.

5.3 Typical Distribution of Maximum Distance Change Values

We first show that much of the change in distance values is *large for a small number of node pairs* in the real data sets tested. Figure 3 shows the distribution of the top 1000 distance change values for each of the five *DBLP* snapshots. We plot the distance change values on the Y-axis, whereas the rank of the node pair (with greatest change) is illustrated on the X-axis. Figure 4 shows the same results for the *IMDB* data set for snapshot changes across the years 1950-1951 and 1951-1952. We note that the distribution of the change values is quite skewed. Only the top few distance change values are very large. This suggests that there are only a few node pairs across which most of the changes were concentrated. This was a recurring theme through the real data sets encountered. The rarity of source nodes reflecting high changes makes it possible to discover them with the use of ranking methods. In the next subsection, we will examine this issue from the perspective of specific nodes (as opposed to node pairs).

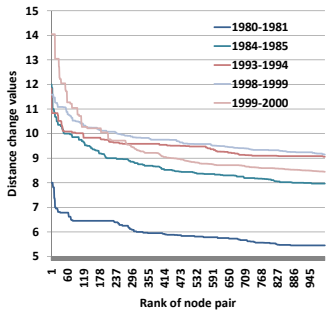


Fig. 3. Variation of shortest path distance changes for *DBLP*

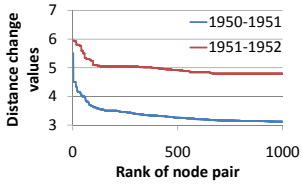


Fig. 4. Variation of shortest path distance changes for *IMDB*

5.4 Contributions by Individual Nodes

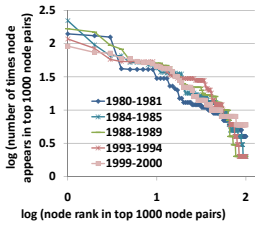
Next, we examine the *involvement of specific nodes* in these pairs of nodes between which most of the distance change is concentrated. Table 4 shows the number of unique nodes in top 1000 maximum distance change node pairs. The results show that a small number of nodes ($\sim$250 on the average, and not increasing significantly with graph size) are a part of the top 1000 maximum distance change pairs. This provides evidence that the ranking approach should approximate the top-k maximum shortest distance change node pairs well, because most of the distances are concentrated in a few nodes.

The trends in this distribution are also illustrated graphically in Figures 5(a) and 5(b) respectively.

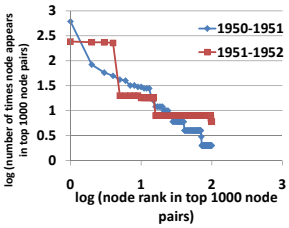
Table 4. Number of nodes in top 1000 maximum distance change node pairs

DBLP Snapshots	#nodes
1980-1981	294
1984-1985	267
1988-1989	112
1993-1994	150
1999-2000	179

IMDB Snapshots	#nodes
1950-1951	343
1951-1952	152



(a) DBLP



(b) IMDB

Fig. 5. A few nodes are responsible for the top few maximum distance change node pairs

5.5 Accuracy Using the Incidence Algorithm and Selective Expansion

Table 5 and Table 6 show the accuracy of our Incidence Algorithm for the *DBLP* and *IMDB* graphs respectively. Each row shows the accuracy for a particular data set over different values of k. Note that even without any selective expansion, the algorithm performs quite well and almost always determines the top five node pairs accurately.

Table 5. Accuracy of Incidence Algorithm (*DBLP*)

Snapshots	K=1	K=5	K=10	K=50	K=100	K=500
1980-1981	1	0.8	0.8	0.8	0.86	0.638
1984-1985	1	1	0.9	0.92	0.8	0.776
1988-1989	1	1	1	0.76	0.84	0.784
1993-1994	1	1	1	1	0.76	0.734
1999-2000	1	1	0.8	0.62	0.69	0.86

Table 6. Accuracy of Incidence Algorithm (*IMDB*)

Snapshots	K=1	K=5	K=10	K=50	K=100	K=500
1950-1951	1	1	1	1	0.93	0.982
1951-1952	1	1	1	0.94	0.8	0.57

Table 7 shows the accuracy of our Incidence Algorithm with selective seed set expansion for the *DBLP* graphs. Note that selective expansion improves accuracy significantly compared to the Incidence Algorithm, and the difference is especially significant for large values of k.

Table 7. Accuracy of the Incidence Algorithm with selective expansion (*DBLP*)

Snapshots	K=1	K=5	K=10	K=50	K=100	K=500
1980-1981	1	0.8	0.8	0.8	0.88	0.84
1984-1985	1	1	1	1	1	1
1988-1989	1	1	1	0.76	0.84	0.896
1993-1994	1	1	1	1	0.76	0.734
1999-2000	1	1	1	0.8	0.69	0.86

5.6 Accuracy of Edge Importance Numbers

We note that the edge importance numbers are *estimated* in order to design the change detection algorithm. For the change detection algorithm to work well, the edge importance numbers must also be estimated accurately. Therefore, in this section we will show that this intermediate result on importance numbers is estimated accurately as well. Later, we will show the final quality of the algorithm with the use of these importance numbers. We note that the precise value of the importance numbers can be estimated accurately, provided that we are willing to spend the time required to run the shortest path algorithm from all the nodes. These precise values are compared with the estimated values of the importance numbers. In order to do so, we compute the importance numbers with a varying level of sampling. Specifically, we vary the number of source nodes from 100 to 1000 and also the number of shortest path trees per source node as 100 and 1000. We report comparisons of some of these with the precise values of the importance numbers.

In our algorithm, the importance number was estimated in two ways: (a) The source nodes were sampled randomly from the entire graph (b) The graph was partitioned with the METIS graph partitioning algorithm [16] into a number of different partitions, which was varied from 100 to 500. A node is then randomly selected from each partition as a source node for edge importance number computation.

We note that only the top few important edges are used in the computation process. Therefore, we are mostly interested in the convergence of the importance numbers of the top few edges. Therefore, we will show the results for the top 100 most important edges. The results are illustrated in Figure 6. Here, we plot the edge importance numbers on Y-axis for each of the edges on the X-axis. Note that all of the curves follow the trend which is quite similar to that of the curve corresponding to precise computations (all_500). It is evident from the results that the importance number values converge quite well for the top few edges. This is especially true, when the clustering-based algorithm is used for the sampling process.

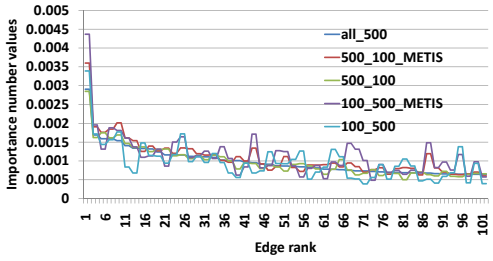


Fig. 6. Convergence of the top 100 edges

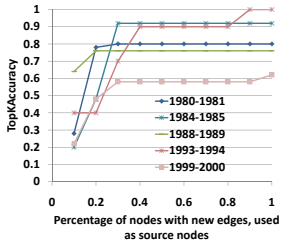


Fig. 7. Performance of CBRP for k=50 (*DBLP*). Nodes with new edges $\approx 20\%$ of total nodes

5.7 Accuracy Comparison of Ranking Algorithms

We note that a ranking algorithm is effective, if we can run Dijkstra algorithm from a very small number of top ranked nodes and still achieve good accuracy for the maximum shortest path distance change node pairs. We executed each of our ranking algorithms for the different data sets, for values of k ranging from 1 to 500. We varied the

number of source nodes used for Dijkstra runs in order to test the sensitivity. An example of such a curve is shown in Figure 7, in which we show the curves for k=50 for the *DBLP datasets* using the *CBRP* ranking algorithm. On the X-axis, we use the percentage of nodes from the original graph (from all nodes having at least one new edge incident on them), which are used as a source node for Dijkstra runs. We note that the area under such a curve is good proxy for the effectiveness of our algorithm, because a more effective algorithm would provide a higher accuracy for a small percentage of samples nodes, and provide a considerable amount of "lift". Therefore, we can compute the average area under the curve as a measure of the goodness of the algorithm. For importance number estimation, we used the METIS-based selection of 100 random nodes and 100 random shortest path trees per source node for all the graphs except for the road network. In the case of the *Ontario RN* data set, we used 20 random nodes and just one shortest path tree per source node as the road network is quite sparse.

Figure 8 shows[11] the accuracy comparisons of our different algorithms for the *DBLP data set*. In addition, we added a few baselines. Specifically, the algorithm labeled as *Random* refers to the case were nodes were randomly chosen from the entire graph for running the Dijkstra algorithm. The algorithm labeled as *RandomNWNE* refers to the algorithm in which the nodes were randomly chosen only from the set of nodes on which new edges were incident between two snapshots. We ran each of the algorithms ten times and reported the average values in order to provide statistical stability to the results. From the results, it is evident that the most simplistic algorithm (or *EWBR* algorithm), which considers only the weight of the new edges incident on a node, performs quite poorly. The algorithm which uses importance numbers (or *INBR* algorithm) turns out to be much more accurate. However, the algorithm which considers change in importance numbers (or *INCBR* algorithm) as well is more effective than either of the previous two algorithms. However, the most superior performance is achieved by the clustering based algorithm (*CBR*). It is interesting to see that the initial selection of the nodes with the use of graph partitioning in *CBRP* provides marginal improvement in accuracy over the variant *CBR* which does not.

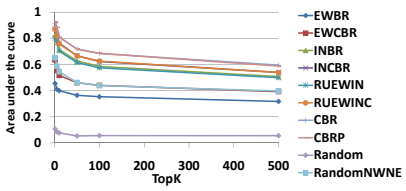


Fig. 8. Accuracy Comparison of Ranking Algorithms:averaged across five pairs of *DBLP* snapshots

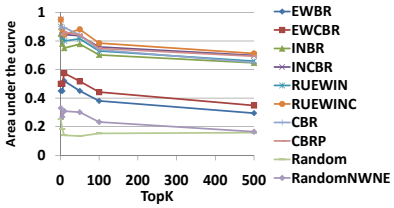


Fig. 9. Accuracy Comparison of Ranking Algorithms (*IMDB*): averaged across two pairs of snapshots

Figures 9 and 10 shows the accuracy comparisons of our different algorithms for the *IMDB* and the *Ontario RN* data sets. The results are quite similar to the previous case with some exceptions as noted well.

One difference is that for graphs such as the road network, our clustering based approach does not work well. This is because a number of the edge changes involve

[11] The acronym for each algorithm in the figures may be found in the section in which these algorithms were introduced.

new *nodes*. The impact of such nodes is hard to estimate with the use of the clustering algorithm.

5.8 Time Comparison of Ranking Algorithms

In this section, we will show the time-comparison of the different ranking algorithms. We note that each ranking algorithm provides a tradeoff between accuracy and efficiency, since the ranking process provides an *order* to the sampling of these nodes. By picking more nodes along this order, it is possible to improve the effectiveness of the algorithm at the expense of efficiency. In this section, we will explore this tradeoff. Results for the *DBLP* data set (1988-1989) are illustrated in Figure 11. We used $k = 50$ to test the accuracy.

Notice that the individual points on each curve are plotted at intervals of 5% of the nodes with new edges. Finally, all the algorithms reach a maximum accuracy of 0.76 reachable by our Incidence Algorithm on this data set for $k = 50$. The importance number based algorithms (*INBR* and *INCBR*) take some time for the initial estimation of the importance numbers. *INCBR* computes importance numbers for both the old and new snapshots and so its initialization time is approximately twice that of *INBR* which estimates importance numbers for the new snapshot only. The same relationship is true between the *RUEWIN* and *RUEWINC* algorithms as well. The clustering based algorithms (*CBR* and *CBRP*) require some additional time for the initial Dijkstra runs from the representative nodes of the clusters and for finding the nearest cluster for all other nodes with new edges. The *CBRP* algorithm requires some extra time for the initial selection of the nodes with the use of graph partitioning.

We notice that the clustering based ranking technique is the most effective from an overall perspective, and is able to provide almost the same amount of accuracy with the use of just 10% of the time required by the Incidence Algorithm. This is because the ranking process is superior to the incidence approach from an algorithmic design perspective, especially when most of the changes are concentrated in a few nodes.

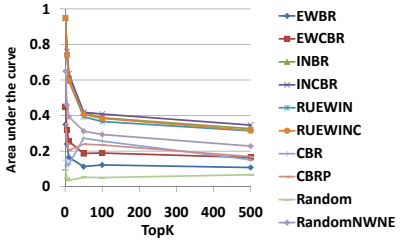


Fig. 10. Accuracy Comparison of Ranking Algorithms (Road network)

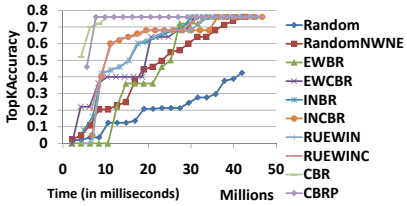


Fig. 11. Time vs Accuracy Comparison of Ranking Algorithms (*DBLP*)

5.9 Case Studies

In this section, we provide a number of insights gained from exploration of the maximum distance change pairs. One interesting observation was that *highly ranked nodes often also had a high level of change in the structural centrality*. For the *DBLP* dataset, Table 8 shows the authors that appeared the maximum number of times in the top 1000 node pairs with maximum shortest path distance change. Such authors are the ones

who never published with popular authors before, and were therefore at the fringes of the co-authorship graph. However, their (or their neighbor's) subsequent participation in publications with popular authors resulted in increasing their structural centrality. In some cases, these were authors who were working in a single research area who recently started collaborating with authors in other research areas.

For example, let us consider the node (author) *Nuno F. Paulino*. The author published with two other authors in 1993. The corresponding publications were also early publications for these authors. In 1994, Nuno published a paper with five other authors including *Gabor C. Temes* who had been publishing since 1971. As a result of this new paper in 1994, Nuno became more central in the graph, and also appeared in the list of top ranking nodes which were among the sources of greatest change in the graph.

Table 8. Nodes with Greatest Change (*DBLP*)

Graphs	Author
1980-1981	Seppo Sippu
1984-1985	W. Eric L. Grimson
1988-1989	Michael Walker
1993-1994	Nuno F. Paulino
1999-2000	R. Böhm

Table 9. Author-pairs with Greatest Change

Graphs	Author1	Author2
1980-1981	B. Krishnamoorthi	Reginald Meeson
1984-1985	W. Eric L. Grimson	Zachary C. Fluhr
1988-1989	Michael Walker	Cees J. A. Jansen
1993-1994	A. D. Bray	Michael Smyth
1999-2000	Jitka Dupacová	Tiecheng Yan

We also present node pairs with greatest change over different graph snapshots in Table 9. These are typically pairs of authors who belong to two different research areas but recently published a paper together, or published with the same common cluster of nodes. E.g., the nodes *Tiecheng Yan* and *Jitka Dupacová* were typically on the fringes of the co-authorship graph. However, *Tiecheng Yan* published a paper with *Stein W. Wallace* in 1993, and one with Jitka Dupacová in 2000. While the two authors Jitka and Tiecheng were originally located on different corners of the graph, this authorship behavior brought them close to one another.

The results for the *IMDB data set* are illustrated in Table 10. The *IMDB* graph was quite dense with many cliques. This is because many actors may work in the same movie, and this adds a large clique of edges in the graph. For example, *Sanders Kerry (II)* acted as a correspondent for the TV series "Today" in 1952. IMDB lists 938 people as related to this TV series. Furthermore, in 1947, he was involved in the "Meet the Press" show. These factors, resulted in his becoming a node with a large change in centrality. This was also reflected in the fact that it was identified as a top-ranked change node.

The results for *Ontario Road Network* are illustrated in Tables 11 and 12. Locations[12] in Table 11 are marked on the map (Google Maps[13]) in Figure 12 (left). From the results, we can see that the shortest path distance between many locations changed in the area situated in the south west part of Algonquin Provincial Park. We looked back at the road networks and discovered that this happened because of the construction of an extension to the Bear Lake Road during the time period from 2005 to 2008. We mark

[12] http://tinyurl.com/mg-mapOne

[13] http://maps.google.com

Table 10. Star pairs with the highest change in shortest path distance

Graphs	Actor1	Actor2
1950-1951	Niculescu-Bruna, Ion	Conabie, Gheorghe
1950-1951	Onea, Arion	Conabie, Gheorghe
1950-1951	Niculescu-Bruna, Ion	Scarlatescu, N.
1951-1952	Sanders, Kerry (II)	Hoffman, Dustin
1951-1952	Sanders, Kerry (II)	Gordon, Mark (I)

the corresponding area[14] in Figure 12 (right). Here points A, E correspond to column 3 of Table 12 and points B, C and D correspond to column 2 of Table 12.

Table 11. Nodes with greatest change over time period '05-'08

Longitude,Latitude	Location
78.7559W,45.3429N	Clayton Lake, near Dorset
78.7343W,45.3464N	Bear Lake Road, Near Livingstone Lake
78.7024W,45.3112N	Kawagama Lake, near Bear island
78.667W,45.3273N	Near Kimball Lake
78.6956W,45.3216N	Near Kawagama Lake

Table 12. Loc pairs with greatest change ('05-'08)

Location1	Location2
78.704W,45.3321N	78.692W,45.3173N
78.6995W,45.3222N	78.692W,45.3173N
78.6995W,45.3222N	78.692W,45.3173N
78.7033W,45.3347N	78.6956W,45.3216N
78.704W,45.3321N	78.692W,45.3173N

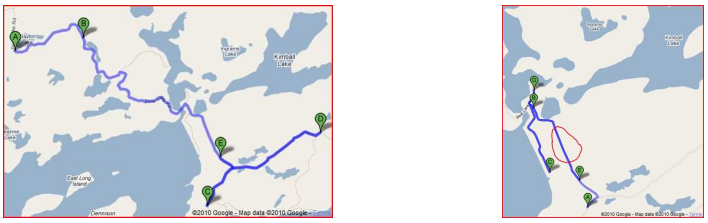


Fig. 12. Representation of locations in Table 11 and 12

From the above case studies, it is evident that our approach can discover interesting node pairs for which shortest path distance change was maximum. Quite often these changes are because of the combinatorial effect of multiple new edges that come in the network. Our algorithms can also discover the most critical edges which result in a significant part of the distance changes. This also helps provide a better understanding of the causality of the changes in the distances.

6 Conclusions and Future Work

In this paper, we presented several fast algorithms to compute top-k node pairs with the greatest evolution in shortest path distances. We experimented using large graphs and showed the efficiency and scalability of our methods. We observed that the change in edge importance number and clustering based methods work quite well for the task. One advantage of the approach is that it was able to create effective results over extremely large data sets in which it was not possible to use traditional methods efficiently. In future work, we will perform investigation of specific structural aspects of graphs which are related to considerable distance change. We will also exploit these algorithms for a variety of event-detection applications in networks.

7 Acknowledgements

Research was sponsored in part by the U.S. National Science Foundation under grant IIS-09-05215, and by the Army Research Laboratory under Cooperative Agreement Number W911NF-09-2-0053 (NS-CTA). The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

[14] http://tinyurl.com/mg-mapTwo

We thank Geography Div, Stats Canada, 2005/ 2008 Road Network File (RNF), 92-500-XWE/XWF. We thank the anonymous reviewers for their insightful comments.

References

1. C. C. Aggarwal (ed.), Social Network Data Analytics, *Springer*, 2011.
2. R. K. Ahuja, T. L. Magnanti, J. B. Orlin. Network Flows: Theory, Algorithms, and Applications. *Prentice Hall*, 1993.
3. R. K. Ahuja, J. B. Orlin, S. Pallottino, M. G. Scutella. Dynamic shortest paths minimizing travel times and costs. *Networks*, 41:205, 2003.
4. M. Henzinger, V. King Maintaining minimum spanning forests in dynamic graphs. *SIAM Journal on Computing*, 31(2):364-374, 2001.
5. L. Akoglu, M. McGlohon, C. Faloutsos. oddball: Spotting Anomalies in Weighted Graphs. *PAKDD*, 410-421, 2010.
6. I. Chabini, S. Ganugapati. Parallel algorithms for dynamic shortest path problems. *International Transactions in Operational Research*, 9(3):279–302, 2002.
7. D. Chakrabarti, R. Kumar, A. Tomkins. Evolutionary clustering. In *KDD*, 554–560, 2006.
8. D. Chakraborty, G. Chakraborty, N. Shiratori. A dynamic multicast routing satisfying multiple qos constraints. *Int. J. Network Management*, 13(5):321–335, 2003.
9. B. V. Cherkassky, A. V. Goldberg, C. Silverstein. Buckets, heaps, lists, and monotone priority queues. In *SODA*, 1997.
10. E. Cohen. Estimating the size of the transitive closure in linear time. *FOCS*, 1994.
11. E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959.
12. P. Doreian, F. Stokman *Evolution of social networks*. Gordon and Breach Publishers 1997.
13. W. Eberle, L. Holder. Discovering structural anomalies in graph-based data. In *ICDMW*, pp. 393–398, 2007.
14. R. W. Floyd. Algorithm 97: Shortest path. *Commun. ACM*, 5(6):345, 1962.
15. D. B. Johnson. Efficient algorithms for shortest paths in sparse networks. *J. ACM*, 24(1):1–13, 1977.
16. G. Karypis V. Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM J. Sci. Comput.*, 20(1):359–392, 1998.
17. J. Leskovec, J. Kleinberg, C. Faloutsos. Graph evolution: Densification and shrinking diameters. *ACM TKDD*, 1(1):2, 2007.
18. P. Papadimitriou, A. Dasdan, H. Garcia-Molina. Web graph similarity for anomaly detection (poster). In *WWW*, pp. 1167–1168, 2008.
19. L. Roditty, U. Zwick. On dynamic shortest paths problems. In *ESA*, 580–591, 2004.
20. A. Shimbel. Structural parameters of communication networks. *Bulletin of Mathematical Biology*, 15:501–507, 1953.
21. Stephen Warshall. A theorem on boolean matrices. *J. ACM*, 9(1):11–12, 1962.
22. S. Zhu, G. M. Huang. A new parallel and distributed shortest path algorithm for hierarchically clustered data networks. *IEEE Trans. Parallel Dist. Syst.*, 9(9):841–855, 1998.
23. M. E. J. Newman, M. Girvan. Finding and evaluating community structure in networks. *Physical Review*, E 69, 2004.
24. U. Brandes. A faster algorithm for betweenness centrality. *Journal of Mathematical Sociology*, 25(2):163–177, 2001.
25. B. Ding, J. X. Yu, L. Qin. Finding time-dependent shortest paths over large graphs. *EDBT*, 205–216, 2008.
26. H. D. Chon, D. Agrawal, A. E. Abbadi. FATES: Finding A Time Dependent Shortest path. *Mobile Data Management*, 165–180, 2003.