Discriminative Distance-Based Network Indices, Link Prediction and the Tiny-World Property

Mostafa Haghir Chehreghani  
LTGI, Télécom ParisTech  
Université Paris-Saclay  
Paris 75013  
mostafa.chehreghani@gmail.com  

Albert Bifet  
LTGI, Télécom ParisTech  
Université Paris-Saclay  
Paris 75013  
albert.bifet@telecom-paristech.fr  

Talel Abdessalem  
LTGI, Télécom ParisTech  
Université Paris-Saclay  
Paris 75013  
talel.abdessalem@telecom-paristech.fr  

ABSTRACT
Distance-based indices, including closeness centrality, average path length, eccentricity and average eccentricity, are important tools for network analysis. In these indices, the distance between two vertices is measured by the length of shortest paths between them. However, this measure has shortcomings. A well-studied shortcoming is that extending it to disconnected graphs (and also directed graphs) is controversial. The second shortcoming is that when this measure is used in real-world networks, a huge number of vertices may have exactly the same closeness/eccentricity scores. This restricts the applicability of these indices as they cannot distinguish vertices. The third shortcoming is that in many applications, the distance between two vertices not only depends on the length of shortest paths, but also on the number of shortest paths between them. In this paper, first we develop a new distance measure between vertices of a graph that yields discriminative distance-based centrality indices. This measure is proportional to the length of shortest paths and inversely proportional to the number of shortest paths. We present algorithms for exact computation of the proposed discriminative indices. Second, we develop randomized algorithms that precisely estimate average discriminative path length and average discriminative eccentricity and show that they give \((\epsilon, \delta)-approximations of these indices \((\epsilon \in \mathbb{R}^+ \text{ and } \delta \in (0, 1))\). Third, we perform extensive experiments over several real-world networks from different domains. In our experiments, we first show that compared to the traditional indices, discriminative indices have usually much more discriminality. Then, we show that our randomized algorithms can very precisely estimate average discriminative path length and average discriminative eccentricity, using only a few samples. Then, we show that real-world networks have usually a tiny average discriminative path length, bounded by a constant (e.g., 2). We refer to this property as the tiny-world property. Fourth, in order to better motivate the usefulness of our proposed distance measure, we present a novel link prediction method, that uses discriminative distance to decide which vertices are more likely to form a link in future, and show its superior performance compared to the well-known existing measures.

KEYWORDS
Distance-based network indices, discriminative indices, closeness centrality, average path length, the tiny-world property, link prediction

ACM Reference format:
Mostafa Haghir Chehreghani, Albert Bifet, and Talel Abdessalem. 2016. Discriminative Distance-Based Network Indices, Link Prediction and the Tiny-World Property. In Proceedings of ACM Conference, Washington, DC, USA, July 2017 (Conference’17), 13 pages. DOI: 10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION
An important category of network indices is based on the distance (the length of the shortest paths) between every two vertices in the network. It includes closeness centrality, average path length, vertex eccentricity, average graph eccentricity, etc. Indices in this category have many important applications in different areas. For example, in disease transmission networks, closeness centrality is used to measure vulnerability to disease and infectivity [5]. In routing networks, vertex eccentricity is used to determine vertices that form the periphery of the network and have the largest worst-case response time to any other device [40, 60]. In biological networks, vertices with high eccentricity perceive changes in concentration of their neighbor enzymes or molecules [52].

Using the length of shortest paths as the distance measure has shortcomings. A well-studied shortcoming is that extending it to disconnected graphs (and also directed graphs) is controversial [17, 48, 56, 63]. The other –less studied– shortcoming is that by using this measure, a huge number of vertices may find exactly the same closeness/eccentricity score. For instance, Shun [59] recently reported that around 30% of the (connected) vertices of the Yahoo graph have the same non-zero eccentricity score. Our experiments, reported in Section 7.1, reveal that this happens in many real-world graphs. This restricts the applicability of distance-based indices such as closeness and eccentricity, as they cannot distinguish vertices. For example, when closeness or eccentricity are used for the facility location problem [31], they may not be able to distinguish one location among a set of candidate locations. Finally, in many cases, the distance between two vertices not only depends on the length of shortest paths, but also on the number of shortest paths between them. As a simple example, consider a network of locations where edges are roads connecting the locations. In a facility location problem, given two (or more) candidate locations, we want to choose the one which is more accessible from the rest of the network. Then, we may prefer the location which is slightly farther.
from the rest of the network but has more connections to the location which is closest to the rest of the network. In particular, if two locations have exactly the same distance from the other locations, the one connected to the rest of the network by more roads is preferred.

These observations motivate us to develop a new distance measure between vertices of a graph that yields more discriminative centrality notions. Furthermore, it considers both shortest path length and the number of shortest paths. In this paper, our key contributions are as follows:

- We propose new distance-based network indices, including discriminative closeness, discriminative path length, discriminative vertex eccentricity and average discriminative graph eccentricity. These indices are proportional to the length of shortest paths and inversely proportional to the number of shortest paths. Our empirical evaluation of these notions reveals an interesting property of real-world networks. While real-world graphs have the small-world property which means they have a small average path length bounded by the logarithm of the number of their vertices, they usually have a considerably smaller average discriminative path length, bounded by a constant (e.g., 2). We refer to this property as the tiny-world phenomena.

- We propose algorithms for exact computation of the proposed discriminative indices. We then develop randomized algorithms that precisely estimate average discriminative path length and average discriminative eccentricity and show that they can give \((\epsilon, \delta)\)-approximations of average discriminative length and average discriminative eccentricity of the graph, where \(\epsilon \in \mathbb{R}^+\) and \(\delta \in (0, 1)\).

- We perform extensive experiments over several real-world networks from different domains. First, we examine discriminability of our proposed indices and show that compared to the traditional indices, they are usually much more discriminative\(^1\). Second, we evaluate the empirical efficiency of our simple randomized algorithm for estimating average discriminative path length and show that it can very precisely estimate average discriminative path length, using only few samples. Third, we show that our simple randomized algorithm for estimating average discriminative eccentricity can generate high quality results, using only few samples. This has analogy to the case of average eccentricity where a simple randomized algorithm significantly outperforms more advanced techniques [59].

- In order to better motivate the usefulness of our proposed distance notion in real-world applications, we present a novel link prediction method, that uses discriminative distance to indicate which vertices are more likely to form a link in future. By running extensive experiments over several real-world datasets, we show the superior performance of our method, compared to the well-known existing indices.

\(^1\)Note that having a total ordering of the vertices is not always desirable and by discriminative indices, we do not aim to do so. Instead, we want to have a partial ordering over a huge number of vertices that using traditional distance-based measures, find exactly the same value.

The rest of this paper is organized as follows. In Section 2, preliminaries and necessary definitions related to distance-based indices are introduced. A brief overview on related work is given in Section 3. In Section 4, we introduce our discriminative distance-based indices and discuss their extensions and properties. We present exact algorithms for computing discriminative indices in Section 5. In section 6 we present randomized algorithms for estimating discriminative indices and analyze them. In Section 7, we empirically evaluate discriminability of our indices and the efficiency and accuracy of our randomized algorithms. In Section 8, we present our link prediction method, which works based on our discriminative distance measure, and show its superior performance. Finally, the paper is concluded in Section 9.

2 PRELIMINARIES

In this section, we present definitions and notations widely used in the paper. We assume that the reader is familiar with basic concepts in graph theory. Throughout the paper, \(G\) refers to a graph (network). For simplicity, we assume that \(G\) is a connected, undirected and loop-free graph without multi-edges. Throughout the paper, we assume that \(G\) is an unweighted graph, unless it is explicitly mentioned that \(G\) is weighted. \(V(G)\) and \(E(G)\) refer to the set of vertices and the set of edges of \(G\), respectively. We denote the set of neighbors of a vertex \(v\) by \(N(v)\).

A shortest path (also called a geodesic path) between two vertices \(v, u \in V(G)\) is a path whose length is minimum, among all paths between \(v\) and \(u\). For two vertices \(v, u \in V(G)\), we use \(d(v, u)\), to denote the length (the number of edges) of a shortest path connecting \(v\) and \(u\). We denote by \(\sigma(v, u)\) the number of shortest paths between \(v\) and \(u\). By definition, \(d(v, v) = 0\) and \(\sigma(v, v) = 0\). We use \(\deg(v)\) to denote the degree of vertex \(v\). The diameter of \(G\), denoted by \(\Delta(G)\), is defined as \(\max_{v, u \in V(G)} d(v, u)\). The radius of \(G\) is defined as \(\min_{v \in V(G)} \max_{u \in V(G) \setminus \{v\}} d(v, u)\).

Closeness centrality of a vertex \(v \in V(G)\) is defined as:

\[
C(v) = \frac{1}{|V(G)| - 1} \sum_{u \in V(G) \setminus \{v\}} d(v, u). \tag{1}
\]

Average path length of graph \(G\) is defined as:

\[
\text{APL}(G) = \frac{1}{|V(G)| \times (|V(G)| - 1)} \sum_{v \in V(G)} \sum_{u \in V(G) \setminus \{v\}} d(v, u). \tag{2}
\]

Eccentricity of a vertex \(v \in V(G)\) is defined as:

\[
E(v) = \frac{1}{|V(G)| - 1} \max_{u \in V(G) \setminus \{v\}} d(v, u). \tag{3}
\]

Average eccentricity of graph \(G\) is defined as:

\[
\text{AE}(G) = \frac{1}{|V(G)| \times (|V(G)| - 1)} \sum_{v \in V(G)} \max_{u \in V(G) \setminus \{v\}} d(v, u). \tag{4}
\]

Center of a graph is defined as the set of vertices that have the minimum eccentricity. Periphery of a graph is defined as the set of vertices that have the maximum eccentricity.

3 RELATED WORK

The widely used distance-based indices are closeness centrality, average path length, eccentricity and average eccentricity defined in
Section 2. In all these indices, it is required to compute the distance between every pair of vertices. The best algorithm in theory for solving all-pairs shortest paths is based on matrix multiplication \([65]\) and its time complexity is \(O((|V|)^{2.3727})\). However, in practice breadth first search (for unweighted graphs) and Dijkstra’s algorithm (for weighted graphs with positive weights) are more efficient. Their time complexities are \(O((|V||E|))\) and \(O(|V|(|E|+|V|^2)\log |V|))\), respectively. In the following, we briefly review exact and inexact algorithms proposed for computing closeness and eccentricity.

Closeness centrality and average path length. Eppstein and Wang \([20]\) presented a uniform sampling algorithm that with high probability approximates the inverse closeness centrality of all vertices in a weighted graph \(G\) within an additive error \(\varepsilon \Delta(G)\). Their algorithm requires \(O(\frac{\log |V|}{\varepsilon^2})\) samples and spends \(O(|V|\log |V|+|E|)\) time to process each sample. Brandes and Pich \([11]\) extended this sampler by considering different non-uniform ways of sampling. A similar uniform sampler is presented in \([41]\) to approximate average path length of a graph. Cohen et al. \([16]\) combined the sampling method with the pivoting approach \([15]\) and \([62]\), where pivoting is used for the vertices that are far from the given vertex. In the algorithm of Olsen et al. \([47]\), when closeness centrality of a vertex is computed, the intermediate results are stored and reused for the centrality computation of the other vertices. Okamoto et al. \([46]\) presented an algorithm for ranking top \(k\) highest closeness centrality vertices that runs in \(O(k+|V|\log |V|+|V||E|)\) time. More recently, Bergamini et al. \([6]\) developed an algorithm for finding top \(k\) highest closeness centrality vertices that finds an upper bound on the closeness score of each vertex. It stops when \(k\) vertices are found whose closeness scores are larger than the upper bounds of the others. Worst case time complexity of their algorithm is the same as time complexity of computing closest score of all vertices. There are several extensions of closeness centrality for specific networks. Kang et al. \([29]\) defined closeness centrality of a vertex \(v\) as the approximate average distance from \(v\) to all other vertices in the graph. Tarkowski et al. \([61]\) developed a game-theoretic extension of closeness centrality to networks with community structure.

Eccentricity and average eccentricity. Dankelmann et al. \([54]\) showed that the average eccentricity of a graph is at least \(\frac{9|V||E|}{4}\), where \(\deg_m\) is the minimum degree of the graph. Roditty and Williams \([57]\) developed an algorithm that gives an estimation \(\hat{E}(v)\) of the eccentricity of vertex \(v\) in an undirected and unweighted graph, such that \(\frac{1}{2}E(v) \leq \hat{E}(v) \leq \frac{3}{2}E(v)\). Time complexity of the algorithm is \(O(|E(G)|\sqrt{|V(G)|}\log |V(G)|)\). Takes and Kosters \([60]\) presented an exact eccentricity computation algorithm, based on lower and upper bounds on the eccentricity of each vertex of the graph. They also presented a pruning technique and showed that it can significantly improve upon the standard algorithms. Chechik et al. \([12]\) introduced an \(O(|E(G)|\log |E(G)|^{3/2})\) time algorithm that gives an estimate \(\hat{E}(v)\) of the eccentricity of vertex \(v\) in an undirected and weighted graph, such that \(\frac{2}{3}E(v) \leq \hat{E}(v) \leq E(v)\). Shun \([59]\) compared shared-memory parallel implementations of several average eccentricity approximation algorithms. He showed that in practice a two-pass simple algorithm significantly outperforms more advanced algorithms such as \([57]\) and \([12]\).

4 DISCRIMINATIVE DISTANCE-BASED INDICES

In this section, we present the family of discriminative distance-based indices. First in Section 4.1, we introduce the indices and discuss their generalizations and properties. Then in Section 4.2, we present some intuitions behind them.

4.1 Indices

The first notion is discriminative closeness centrality. Similar to closeness centrality, discriminative closeness is based on the length of shortest paths between different vertices in the graph. However, unlike closeness centrality, discriminative closeness centrality considers the number of shortest paths, too. For a vertex \(v \in V(G)\), discriminative closeness of \(v\), denoted with \(DC(v)\), is formally defined as follows:

\[
DC(v) = \frac{1}{|V(G)|-1} \sum_{u \in V(G)} \frac{d(v, u)}{\sigma(v, u)}
\]

As mentioned earlier in the Introduction section and as we will present some cases in Section 4.2, the rationale behind discriminative closeness centrality is that if vertex \(v\) has the same distance from vertices \(u_1\) and \(u_2\) but \(\sigma(v, u_1) > \sigma(v, u_2)\), in many applications we can say that \(u_1\) is closer than \(u_2\) to \(v\). In Equation 2 of Section 2, if closeness centrality is replaced by discriminative closeness centrality defined in Equation 5, we get average discriminative path length of \(G\), defined as follows:

\[
ADPL(G) = \frac{1}{|V(G)|} \times (|V(G)|-1) \sum_{v \in V(G)} \sum_{u \in V(G), \{v\}} d(v, u) / \sigma(v, u)
\]

In a similar way, discriminative eccentricity of a vertex \(v \in V(G)\), denoted with \(DE(v)\), is defined as follows:

\[
DE(v) = \frac{1}{|V(G)|-1} \max_{u \in V(G), \{v\}} d(v, u) / \sigma(v, u)
\]

Finally, average discriminative eccentricity of \(G\) is defined as follows:

\[
ADEC(G) = \frac{1}{|V(G)|} \times (|V(G)|-1) \sum_{v \in V(G)} \max_{u \in V(G), \{v\}} d(v, u) / \sigma(v, u)
\]

All these notions are based on replacing distance by discriminative distance, defined as follows. For \(v, u \in V(G)\), discriminative distance between \(v\) and \(u\), denoted with \(dd(v, u)\), is defined as \(\frac{d(v, u)}{\sigma(v, u)}\). We define discriminative diameter and discriminative radius of \(G\) respectively as follows:

\[
DD(G) = \max_{v \in V(G)} \max_{u \in V(G), \{v\}} \frac{d(v, u)}{\sigma(v, u)}
\]

\[
DR(G) = \min_{v \in V(G)} \max_{u \in V(G), \{v\}} \frac{d(v, u)}{\sigma(v, u)}
\]

Finally, we define discriminative center of a graph as the set of vertices that have the minimum discriminative eccentricity; and discriminative periphery of a graph as the set of vertices that have the maximum eccentricity.
Note that path-based indices such as betweenness centrality [13] (and its generalizations such as group betweenness centrality [21] and co-betweenness centrality [24]) consider the number of shortest paths that pass over a vertex. However, betweenness centrality does not consider the shortest path length and it is used as an indicator of the amount of control that a vertex has over shortest paths in the network. Some variations of betweenness centrality, such as length-scaled betweenness centrality and linearly scaled betweenness centrality [10], are more similar to our proposed notion. However, they still measure the amount of control that a vertex has over shortest paths, but give a weight (which is a function of distance) to the contribution of each shortest path. In our proposed notion, the number of shortest paths passing over a vertex does not always contribute in the centrality of the vertex. Indices such as Katz centrality [30] and personalized PageRank [26] consider both the length and the number of paths between two vertices. However, they have shortcomings. For example, Katz centrality is proportional to both the length and the number of paths. Furthermore, it considers all paths. This makes it inappropriate for the applications where the concept of shortest paths is essential. This index is mainly used in the analysis of directed acyclic graphs. The other index that may have some connection to our discriminative indices is clustering coefficient [64]. Both clustering coefficient and discriminative indices are sensitive to the local density of the vertices, however, they have different goals. While clustering coefficient aims to directly reflect the local density, discriminative indices aim to take into account density of different regions of the graph, when computing distances.

Generalizations. We can consider two types of generalizations of Equations 5-10. In the first generalization, in the denominator of the equations, instead of using the number of shortest paths, we may use the number of a restricted class of shortest paths, e.g., vertex-disjoint shortest paths, edge-disjoint shortest paths etc. In the second generalization, instead of directly using distances and the number of shortest paths, we may introduce and use functions $f$ and $g$, defined respectively on the length and the number of shortest paths. Then, by changing the definitions of $f$ and $g$, we can switch among different distance-based notions. For example, for any two vertices $v, u \in V(G)$, if $f(d(v, u))$ and $g(\sigma(v, u))$ are respectively defined as $d(v, u)$ and 1, we will have the traditional distance-based indices introduced in Section 2. If $f(d(v, u)) = 1$ and $g(\sigma(v, u)) = \sigma(v, u)$, we will have harmonic closeness centrality [56] defined as follows:

$$HC(v) = \frac{1}{|V(G)| - 1} \sum_{u \in V(G)} \frac{1}{d(v, u)}.$$  

Then, someone may define discriminative harmonic closeness centrality of vertex $v$ as:

$$DHC(v) = \frac{1}{|V(G)| - 1} \sum_{u \in V(G)} \frac{\sigma(v, u)}{d(v, u)},$$  

(11)

where $f(d(v, u))$ and $g(\sigma(v, u))$ are respectively defined as $\frac{1}{d(v, u)}$ and $\frac{1}{\sigma(v, u)}$.

Disconnected or directed graphs. When the graph is disconnected or directed, it is possible that there is no (shortest) path between vertices $v$ and $u$. In this case, $d(v, u) = \infty$ and $\sigma(v, u) = 0$, hence, $d(v, u) / \sigma(v, u)$ is undefined. For closeness centrality, when $d(v, u) = \infty$, a first solution is to define $d(v, u)$ as $|V(G)|$. The rationale is that in this case $d(v, u)$ is a number greater than any shortest path length. We can use a similar technique for discriminative distance: when there is no path from $v$ to $u$, we define $d(v, u)$ as $|V(G)|$ and $\sigma(v, u)$ as 1. This discriminative distance will be greater than the discriminative distance between any two vertices $v'$ and $u'$ that are connected by a path from $v'$ to $u'$. The second solution suggested for closeness centrality is to use harmonic centrality [56]. As stated in Equation 11, this can be applied to discriminative closeness, too. When $d(v, u) = \infty$, Equation 11 yields $\frac{1}{\sigma(v, u)}$, which is conventional to define as 0.

A property. A nice property of shortest path length is that for vertices $v, u, w \in V(G)$ such that $w$ is on a shortest path between $v$ and $u$, the following holds: $d(v, u) = d(v, w) + d(w, u)$. This property is useful in e.g., designing efficient distance computation algorithms. This property does not hold for discriminative distance as $d(v, u)$ can be less than or equal to $d(v, w) + d(w, u)$. An example is presented in Figure 1. However, we believe this is not a serious problem. The reason is that more than shortest path length that satisfies the above mentioned property, discriminative distance is based on the number of shortest paths, which satisfies the following property: $\sigma_{vw}(v, u) = \sigma(v, w) \times \sigma(w, u)$, where $\sigma_{vw}(v, u)$ is the number of shortest paths between $v$ and $u$ that pass over $w$. As we will discuss in Section 5, these two properties can help us to design efficient algorithms for computing discriminative distance-based indices.

4.2 Intuitions

In several cases, the distance between two vertices in the graph not only depends on their shortest path length, but also (inversely) on the number of shortest paths they have. In the following, we discuss some cases.

Time and reliability of traveling. A key issue in transportation and logistics is to estimate the traveling time and route reliability between two given points $A$ and $B$ [8, 55]. The time and reliability of traveling from $A$ to $B$ depend on the structure of the road network and also on the stochastic factors such as weather, traffic incidents etc. One of the factors that depends on the network structure is the number of ways that someone can travel from $A$ to $B$. Having several ways to travel from $A$ to $B$, on the one hand, increases the reliability of traveling, as in the case of failure in one of the ways, $B$ is still reachable from $A$. On the other hand, it decreases traffic
between $A$ and $B$ and as a result, the traveling time. Therefore, taking into account both the length and the number of shortest paths between $A$ and $B$ (in other words, defining the distance between $A$ and $B$ in terms of both the length and the number of shortest paths between $A$ and $B$) can help to better estimate the time and reliability of traveling between $A$ and $B$.

Spread of infections. It is known that infections and contacts rates in a network depend on the community structure of the network and the spread of infections inside a community is faster [3, 38, 53]. Consider vertices $v_1, v_2$ and $v_3$ such that $d(v_1, v_2) = d(v_1, v_3)$, $v_1$ and $v_2$ are in the same community but $v_1$ and $v_3$ are not. An infection from $v_1$ usually spreads to $v_2$ faster than $v_3$, or the probability that (after some time steps) $v_2$ becomes infected by $v_1$ is higher than the probability that $v_3$ becomes infected. Here, to describe the distances between the vertices, our discriminative distance measure is a better notion than the shortest path length. Vertices $v_1$ and $v_2$ that are inside a community are heavily connected and as a result, they usually have many shortest paths between themselves. In contrast, $v_1$ and $v_3$ do not belong to the same community and are not heavily connected, hence, they usually have less shortest paths between themselves. This means $dd(v_1, v_2)$ is smaller than $dd(v_1, v_3)$, which is consistent with the infection rate.

5 EXACT ALGORITHMS

In this section, we present the $\text{DCC}^2$ algorithm for computing discriminative closeness centrality of all vertices of the network and show how it can be revised to compute the other discriminative indices. Algorithm 1 shows the high level pseudo code of the algorithm. $\text{DCC}$ is an iterative algorithm where at each iteration, discriminative closeness of a vertex $v$ is computed. This is done by calling the $\text{ShortestPathDAG}$ method for $v$. $\text{ShortestPathDAG}$ consists of two phases. During the first phase, the distances between $v$ and all other vertices in the graph are computed. If $G$ is unweighted, this is done by a breadth-first search starting from $v$. Otherwise, if $G$ is weighted with positive weights, this is done using Dijkstra’s algorithm [18]. During the second phase, the number of shortest paths between $v$ and every vertex in the graph is computed. This is done as follows.

- For any vertex $u \in V(G) \setminus \{v\}$ which is a neighbor of $v$, $\sigma(v, u)$ is set to 1. Note that our graphs are simple, where multi-edges and self-loops are not allowed.
- For any vertex $u \in V(G) \setminus \{v\}$ which is not a neighbor of $v$, as Lemma 3 of [9] suggests, $\sigma(v, u)$ is calculated as $\sum_{w \in P_{v}(u)} \sigma(v, w)$, where the set $P_{v}(u)$ is $\{w \in N(u)|w$ is on a shortest path from $v$ to $u\}$, and it is computed during the first phase.

Complexity analysis. For unweighted graphs, each iteration of the loop in Lines 7-12 of method $\text{DCC}$ takes $O(|E(G)|)$ time. Note that on the one hand, for unweighted graphs, Line 6 of method $\text{ShortestPathDAG}$ takes $O(|E(G)|)$ time. On the other hand, for unweighted graphs, for each vertex $v \in V(G)$ Lines 17-21 of

```
\begin{algorithm}
\caption{High level pseudo code of the algorithm of computing discriminative closeness scores.}
1: \text{DCC}
2: \textbf{Input.} A network $G$
3: \textbf{Output.} Discriminative closeness centrality of vertices of $G$
4: \textbf{for each} vertex $v \in V(G)$ \textbf{do}
5: \quad $DC[v] \leftarrow 0$
6: \textbf{end for}
7: \textbf{for each} vertex $v \in V(G)$ \textbf{do}
8: \quad $D, N \leftarrow \text{ShortestPathDAG}(G, v)$
9: \quad \textbf{for each} vertex $u \in V(G) \setminus \{v\}$ \textbf{do}
10: \quad \quad $DC[u] \leftarrow DC[u] + \frac{1}{|V(G)|-1} \times D[u]$
11: \textbf{end for}
12: \textbf{end for}
13: \textbf{return} $DC$
\end{algorithm}
```

method $\text{ShortestPathDAG}$ takes $O(deg(v))$ time, hence, it takes $O(|E(G)|)$ time for all the vertices. For weighted graphs with positive weights, it takes $O(|E(G)|+|V(G)| \log |V(G)|)$ time. Hence, time complexity of the algorithm for unweighted and weighted graphs (with positive weights) is respectively $O(|V(G)||E(G)|)$ and $O(|V(G)||E(G)|+|V(G)|^2 \log |V(G)|)$.

Computing other indices. The DCC method of Algorithm 1 can be revised to compute average discriminative path length of $G$, discriminative eccentricity of vertices of $G$ and average discriminative eccentricity of $G$.
ADPL(G). After Line 12 of the DCC method of Algorithm 1 (where the DC[\*v] values are already computed), ADPL(G) can be computed as \( \frac{1}{|V(G)|} \sum_{v \in V(G)} DC[v] \).

- DE(v). If Line 10 of the DCC method of Algorithm 1 is replaced by the following lines:

\[
\text{if } \frac{1}{|V(G)|-1} \times \sum_{u \in V(G) \setminus \{v\}} \frac{d(v, u)}{N[u]} > DC[v] \text{ then}
DC[v] \leftarrow \frac{1}{|V(G)|-1} \times \sum_{u \in V(G) \setminus \{v\}} \frac{d(v, u)}{N[u]}
\]

end if

then, the algorithm will compute discriminative eccentricity of the vertices of \( G \) and will store them in the DC array.

- ADE(G). After computing discriminative eccentricity of all vertices of \( G \) and storing them in the DC array, ADE(G) can be computed as \( \sum_{v \in V(G)} DC[v] \).

In a similar way, Algorithm 1 can be revised to compute discriminative diameter and discriminative radius of \( G \). In all these cases, time complexity of Algorithm 1 remains the same.

6 RANDOMIZED ALGORITHMS

As discussed in Section 5, computing average discriminative path length and average discriminative eccentricity is tractable in theory in the sense that they have polynomial time (and space) algorithms. However, the algorithms are computationally expensive in practice, even for mid-size networks. This motivates us to present randomized algorithms that can be performed much faster, at the expense of having approximate results. In this section, we first introduce a randomized algorithm for estimating average discriminative path length and analyze it. Then, we discuss how this algorithm can be revised to estimate average discriminative eccentricity.

Algorithm 2 High level pseudo code of the algorithm of estimating average discriminative path length.

1: RANDOMADPL
2: Input: A network \( G \) and the number of samples \( T \).
3: Output: Estimated average discriminative path length of \( G \).
4: \( \beta \leftarrow 0 \).
5: for each \( t = 1 \) to \( T \) do
6: Select a vertex \( v \in V(G) \) uniformly at random.
7: \( D, N \leftarrow \text{ShortestPathDAG}(G, v) \).
8: \( \beta_t \leftarrow \frac{1}{|V(G)|-1} \times \sum_{u \in V(G) \setminus \{v\}} \frac{D[u]}{N[u]} \).
9: \( \beta \leftarrow \beta + \frac{\beta_t}{T} \).
10: end for
11: \( \beta \leftarrow \frac{\beta}{T} \).
12: return \( \beta \).

6.1 The Algorithm

Algorithm 2 shows the high level pseudo code of the RANDOMADPL algorithm, proposed to estimate average discriminative path length. The inputs of the algorithm are the graph \( G \) and the number of samples (iterations) \( T \). In each iteration \( t \), the algorithm first chooses a vertex \( v \) uniformly at random and calls the \text{ShortestPathDAG} method for \( v \) and \( G \), to compute distances and the number of shortest paths between \( v \) and any other vertex in \( G \). Then, it estimates average discriminative path length of \( G \) at iteration \( t \) as \( \frac{1}{|V(G)|-1} \times \sum_{u \in V(G) \setminus \{v\}} \frac{d(v, u)}{N[u]} \) and stores it in \( \beta_t \). The average of all \( \beta_t \) values computed during different iterations gives the final estimation \( \beta \) of average discriminative path length.

Complexity analysis. For unweighted graphs, time complexity of Algorithm 2 is \( O(T \times |E(G)|) \). For weighted graphs with positive weights, time complexity of Algorithm 2 is \( O(T \times |E(G)| + T \times |V(G)| \log |V(G)|) \).

6.2 Error Guarantee

In the following, we provide an error bound for the estimated value of average discriminative path length. First in Proposition 6.1, we prove that in Algorithm 2 the expected value of \( \beta \) is ADPL(G). Then in Proposition 6.2, we provide an error bound for \( \beta \).

**Proposition 6.1.** In Algorithm 2, expected value of \( \beta_t \)’s (\( 1 \leq t \leq T \)) and \( \beta \) is ADPL(G).

**Proof.** We have:

\[
E[\beta_t] = \sum_{v \in V(G)} \left( \frac{1}{|V(G)|} \times \sum_{u \in V(G) \setminus \{v\}} \frac{d(v, u)}{N[u]} \right)
= ADPL(G),
\]

where \( \frac{1}{|V(G)|} \) comes from the uniform distribution used to choose vertices of \( G \). Then, we have:

\[
E[\beta] = \frac{\sum_{t=1}^{T} E[\beta_t]}{T} = \frac{T \times E[\beta_t]}{T} = ADPL(G).
\]

**Proposition 6.2.** In Algorithm 2, let \( G \) be a connected and undirected graph. For a given \( \epsilon \in \mathbb{R}^+ \), we have:

\[
P(|ADPL(G) - \beta| > \epsilon) \leq 2 \exp \left(-2 \times T \times \left( \frac{\epsilon}{\Delta(G)} \right)^2 \right).
\]

**Proof.** The proof is done using Hoeffding’s inequality [27]. Let \( X_1, \ldots, X_n \) be independent random variables bounded by the interval \([a, b] \), i.e., \( a \leq X_i \leq b \) (\( 1 \leq i \leq n \)). Let also \( \tilde{X} = \frac{1}{n} \sum_{i=1}^{n} X_i \). Hoeffding [27] showed that:

\[
P \left[ |E[\tilde{X}] - \tilde{X}| > \epsilon \right] \leq 2 \exp \left(-2n \left( \frac{\epsilon}{b-a} \right)^2 \right).
\]

On the one hand, for any two distinct vertices \( v, u \in V(G) \), we have: \( d(v, u) \leq \Delta(G) \) and \( \sigma(v, u) \geq 1 \). Therefore, \( \frac{d(v, u)}{\sigma(v, u)} \leq \Delta(G) \) and as a result, \( \beta_t \leq \Delta(G) \) (\( 1 \leq t \leq T \)). On the other hand, for any two distinct vertices \( v, u \in V(G) \), we have: \( \frac{d(v, u)}{\sigma(v, u)} > 0 \). Therefore, \( \beta_t > 0 \), for \( 1 \leq t \leq T \). Note that in Algorithm 2 vertices \( u \) are chosen independently and therefore, variables \( \beta_t \) are independent. Hence, we can use Hoeffding’s inequality, where \( X_i ’s \), \( \tilde{X} \) is \( \beta \), \( \epsilon \) is \( T \), \( a \) is 0 and \( b \) is \( \Delta(G) \). Putting these values into Inequality 13 yields Inequality 12.

Real-world networks have a small diameter, bounded by the logarithm of the number of vertices in the network [64]. This, along
with Inequality 12, yields\(^4\):

\[
P[|ADPL(G) - \beta| > \epsilon] \leq 2 \exp \left( -2 \times T \times \left( \frac{\epsilon}{\log |V(G)|} \right)^2 \right). \tag{14}
\]

Inequality 14 says that for given values \(\epsilon \in \mathbb{R}^+\) and \(\delta \in (0, 1)\), if \(T\) is chosen such that

\[
T \geq \frac{\ln \left( \frac{1}{\delta} \right)}{2\epsilon^2} \left( \log |V(G)| \right)^2,
\]

Algorithm 2 estimates average discriminative path length of \(G\) within an additive error \(\epsilon\) with a probability at least \(\delta\). Our extensive experiments reported in Table 1 show that many real-world networks have a very small discriminative diameter, much smaller than the logarithm of the number of vertices they have. So, we may assume that their discriminative diameter is bounded by a constant \(c\). For such networks, using only \(\frac{\epsilon \log |V(G)|}{2c^2}\) samples, Algorithm 2 can estimate average discriminative path length within an additive error \(\epsilon\) with a probability at least \(\delta\).

### 6.3 Estimating Average Discriminative Eccentricity

Algorithm 2 can be modified to estimate discriminative eccentricity of graph \(G\). To do so, we require to replace Line 8 of the algorithm by the following lines:

```python
for each \(u \in V(G) \setminus \{v\} \) do
    if \(\frac{1}{|V(G)| - 1} \times \frac{D[u]}{|N[u]|} > \beta_t\) then
        \(\beta_t \leftarrow \frac{1}{|V(G)| - 1} \times \frac{D[u]}{|N[u]|}\).
    end if
end for
```

The rest of the algorithm remains unchanged. It can be shown that in this case the expected value of \(\beta\) will be \(ADE(G)\). The proof is similar to the proof of Proposition 6.1. Furthermore, the same error bound presented in Inequality 12 can be shown for the estimated average discriminative eccentricity. This means for the real-world graphs that have a small diameter bounded by the logarithm of the number of vertices, for given values \(\epsilon \in \mathbb{R}^+\) and \(\delta \in (0, 1)\), if \(T\) satisfies Inequality 15, average discriminative eccentricity of the graph can be estimated within an additive error \(\epsilon\) with a probability at least \(\delta\).

### 7 EXPERIMENTAL RESULTS

We perform extensive experiments on real-world networks to assess the quantitative and qualitative behavior of our proposed algorithms. The programs are compiled by the GNU C++ compiler 5.4.0 using optimization level 3. We do our tests over several real-world datasets from different domains, including the dblp0305, dblp0507 and dblp9202 co-authorship networks [7], the facebook-uniform social network [23], the flickr network [44], the gottron-reuters network [36], the petster-friendships network [32], the pics\(_\text{ut}\) network [32], the web-\textit{NotreDame} network [2], the citeulike-\textit{ut} network [19], the epinions network [43] and the wordnet network [22]. All the networks are treated as undirected graphs. When a graph is disconnected, we consider only its largest component. Table 1 summarizes specifications of the largest components of our real-world networks, including their discriminative diameter.

#### 7.1 Empirical Evaluation of Discriminability

We measure discriminability of a centrality notion in terms of its power in assigning distinguished values to the vertices. Hence, for each centrality notion and over each network \(G\), we define discriminability as:

\[
\frac{\text{#distinct centrality scores}}{\text{#vertices of } G} \times 100.
\]

Table 2 reports the results. The second and fourth columns respectively present discriminability of discriminative closeness and discriminability of closeness. A 'higher percentage' means a 'lower discriminability' of the centrality notion. The rightmost column (titled ratio) shows the ratio of the discriminability of discriminative closeness centrality to the discriminability of closeness centrality. The followings can be seen in the table. First, discriminative closeness centrality is always more discriminative than closeness centrality. Second, over datasets such as dblp0305, dblp0507, dblp9202, facebook-uniform and flickr, discriminability of discriminative closeness centrality is significantly larger than discriminability of closeness centrality. In fact, when discriminability of closeness centrality in a network is very low, discriminative closeness centrality becomes significantly more discriminative than closeness centrality. However, when closeness centrality itself is discriminative (e.g., in datasets pics\(_\text{ut}\), web-\textit{NotreDame}, citeulike-\textit{ut}, epinions and wordnet), the difference between discriminability of these two centrality notions is not considerable.

Table 2 also compares running times of computing these two notions. The algorithm of computing closeness centrality and the algorithm of computing discriminative closeness centrality have the same time complexity. However, in practice, to compute discriminative closeness centrality, we require to have extra traverses over the graph (the second phase of the ShortestPathDAG method), to count the number of shortest paths. This makes the algorithm of computing discriminative closeness centrality slower than the algorithm of computing closeness centrality. In our experiments, the algorithm of computing discriminative closeness centrality is between 2 and 3 times slower than the algorithm of computing closeness centrality.

In order to better highlight the discriminability of our notions, in Table 3 we present the number of repetitions of the most repeated closeness and discriminative closeness scores, denoted respectively by \(f_{max}(C)\) and \(f_{max}(DC)\), for the datasets dblp0305, dblp0507, dblp9202 and facebook-uniform. As can be seen in the table, \(f_{max}(C)\) is considerably larger than \(f_{max}(DC)\). This restricts the applicability of closeness centrality. To understand better, consider a well-known application of closeness centrality, called facility location, where given a set of feasible vertices (locations), we want to choose the one which is closest to the rest of the network. If (most of) the given vertices have exactly the same closeness score, it becomes difficult to choose one. In this case, discriminative closeness may be applied to distinguish those vertices that have the same closeness centrality.

\(^4\)Note that in Inequality 14, both \(\beta\) and \(\epsilon\) are in \(\mathbb{R}^+\) and since \(\beta\) and its expected value are not bounded by \((0, 1)\) and they are considerably larger than 0 (and they can be larger than 1), \(\epsilon\) is usually set to a value much larger than 0 (and even larger than 1, such as \(\log |V(G)|\)).
Table 1: Specifications of the largest component of the real-world datasets.

| Dataset          | Link                                                                 | # vertices | # edges | Discriminative diameter |
|------------------|----------------------------------------------------------------------|------------|---------|-------------------------|
| dblp0305         | http://www-kdd.isti.cnr.it/GERM/                                     | 109,045    | 233,962 | 2                       |
| dblp0507         | http://www-kdd.isti.cnr.it/GERM/                                     | 135,116    | 290,364 | 2                       |
| dblp9202         | http://www-kdd.isti.cnr.it/GERM/                                     | 129,074    | 277,082 | 2                       |
| facebook-uniform | http://odysseas.calit2.uci.edu/doku.php/public:online, social_networks | 134,304    | 135,532 | 2                       |
| flickr           | http://konect.uni-koblenz.de/networks/flickrEdges                    | 73,342     | 2,619,711 | 5                      |
| gottron-reuters   | http://konect.uni-koblenz.de/networks/gottron-reuters                | 38,677     | 978,461 | 5                       |
| petster-friendships | http://konect.uni-koblenz.de/networks/petster-friendships-cat       | 148,826    | 5,449,508 | 8                      |
| pics_ut          | http://konect.uni-koblenz.de/networks/pics_ut                        | 82,035     | 2,300,296 | 5                      |
| web-Stanford     | http://snap.stanford.edu/data/web-Stanford.html                       | 255,265    | 2,234,572 | 16                     |
| web-NotreDame    | http://snap.stanford.edu/data/web-NotreDame.html                     | 325,729    | 1,524,589 | 28                     |
| citeulike-ut     | http://konect.uni-koblenz.de/networks/citeulike-ut                   | 153,277    | 2,411,940 | 7                       |
| epinions          | http://konect.uni-koblenz.de/networks/epinions                       | 119,130    | 834,000  | 15                     |
| wordnet          | http://konect.uni-koblenz.de/networks/wordnet-words                  | 145,145    | 656,230  | 15                     |

Table 2: Comparison of closeness and discriminative closeness centralities over different real-world networks.

| Database            | Discriminative centrality | Closeness centrality |
|---------------------|---------------------------|----------------------|
|                     | Discriminability (%)      | Time (sec)           | Discriminability (%) | Time (sec) | Ratio |
|---------------------|---------------------------|----------------------|----------------------|------------|-------|
| dblp0305            | 2.7805                    | 456.825              | 0.0201               | 364.315    | 137.8181 |
| dblp0507            | 2.7013                    | 1129.59              | 0.0155               | 666.957    | 173.8095 |
| dblp9202            | 3.2973                    | 1697.75              | 0.0147               | 750.8      | 224    |
| facebook-uniform    | 5.6178                    | 1477.43              | 0.0446               | 583.903    | 125.75 |
| flickr              | 92.7694                   | 4769.04              | 4.4435               | 2099.38    | 20.8772 |
| gottron-reuters      | 88.9934                   | 1817.05              | 25.8810              | 782.227    | 3.4385 |
| petster-friendships  | 70.0764                   | 3083.74              | 39.2176              | 8515.56    | 1.7868 |
| pics_ut             | 50.5113                   | 6053.71              | 36.3552              | 2059.86    | 1.3893 |
| web-Stanford        | 97.3376                   | 30117.5              | 18.9258              | 15829.8    | 5.1431 |
| web-NotreDame       | 29.9819                   | 28087.4              | 18.5230              | 15879.1    | 1.6186 |
| citeulike-ut        | 45.2540                   | 12897.6              | 30.4546              | 4968.95    | 1.4859 |
| epinions             | 70.0218                   | 5386.93              | 57.0679              | 2143.99    | 2.2269 |
| wordnet             | 58.8907                   | 6532.28              | 51.8770              | 2725.97    | 1.8852 |

Table 3: Comparison of $f_{max}(DC)$ and $f_{max}(C)$.

| Dataset          | $f_{max}(DC)$ | $f_{max}(C)$ | $f_{max}(C)/f_{max}(DC)$ |
|------------------|---------------|--------------|---------------------------|
| dblp0305         | 11712         | 47431        | 4.049                     |
| dblp0507         | 12942         | 58573        | 4.52                      |
| dblp9202         | 12868         | 55297        | 4.29                      |
| facebook-uniform | 1312          | 124126       | 94.60                     |

7.2 Empirical Evaluation of Estimating Average Discriminative Path Length and Average Discriminative Eccentricity

Table 4 presents the results of the empirical evaluation of our proposed randomized algorithm for estimating average discriminative path length. When estimating average discriminative path length or average discriminative eccentricity, we define relative error of the approximation algorithm as:

$$\frac{|\text{exact score} - \text{approximate score}|}{\text{exact score}} \times 100,$$

where exact score and approximate score are respectively the values computed by the exact and approximate algorithms. Sample sizes are expressed in terms of the percentages of the number of vertices of the graph. We examine the algorithm for three sample sizes: 10% of the number of vertices, 1% of the number of vertices and 0.1% of the number of vertices. As can be seen in the table, only a very small sample size, e.g., 0.1% of the number of vertices, is sufficient to have an accurate estimation of average discriminative path length. Over all the datasets, except gottron-reuters, this sample size gives a relative error less than 3%. In particular, relative error in the datasets dblp0305, dblp0507, dblp9202 and facebook-uniform is very low. This is consistent with our analysis presented in Section 6.2 and is due to very small discriminative diameter of these networks.

Table 5 reports the results of the empirical evaluation of our randomized algorithm for estimating average discriminative eccentricity. Similar to the case of average discriminative path length, we test the algorithm for three different sample sizes and our experiments show that only a small sample size, e.g., 0.1% of the number of vertices, can yield a very accurate estimation of average discriminative eccentricity. In our experiments, for the sample size 0.1%, relative
error is always less than 5%. This high accuracy is due to very small discriminative diameter of the networks. Similar to the case of average eccentricity where a simple randomized algorithm significantly outperforms advanced techniques [59], our simple algorithms show very good efficiency and accuracy for estimating average discriminative path length and average discriminative eccentricity.

7.3 The Tiny-World Property

It is well-known that in real-world networks, average path length is proportional to the logarithm of the number of vertices in the graph and it is considerably smaller than the largest distance that two vertices may have in a graph [64]. Our extensive experiments presented in Table 4 reveal that in real-world networks average discriminative path length is much more smaller than the largest discriminative distance \(^1\) that two vertices may have in a graph and it is

\begin{table}[h]
\centering
\begin{tabular}{|c|c|c|c|c|}
\hline
Database & Exact algorithm & Approximate algorithm & Sample size (%) & Relative error (%) \\
\hline
ADPL & 1.9999 & 10 & 0.0016 & 10 & 0.0016 \\
 & & 1 & 0.0016 & 0.1 & 0.0016 \\
\hline
dblp0305 & 1.9999 & 10 & 0.0008 & 1 & 0.0008 \\
 & & 0.1 & 0.0008 & 0.1 & 0.0008 \\
\hline
dblp0507 & 1.9999 & 10 & 0.0015 & 1 & 0.0015 \\
 & & 0.1 & 0.0012 & 0.1 & 0.0012 \\
\hline
dblp9202 & 1.9999 & 10 & 0.0097 & 1 & 0.0099 \\
 & & 0.1 & 0.0102 & 0.1 & 0.0102 \\
\hline
facebook-uniform & 1.9995 & 10 & 2.5639 & 10 & 2.5639 \\
 & & 1 & 0.2887 & 1 & 0.2887 \\
 & & 0.1 & 2.7859 & 0.1 & 2.7859 \\
\hline
citeulike-ut & 0.2787 & 10 & 0.5827 & 10 & 0.5827 \\
 & & 1 & 3.6259 & 1 & 3.6259 \\
 & & 0.1 & 19.3603 & 0.1 & 19.3603 \\
\hline
epinions & 0.6860 & 10 & 0.8221 & 10 & 0.8221 \\
 & & 1 & 0.4398 & 1 & 0.4398 \\
 & & 0.1 & 2.4948 & 0.1 & 2.4948 \\
\hline
web-Stanford & 0.2220 & 10 & 0.4478 & 10 & 0.4478 \\
 & & 1 & 1.8667 & 1 & 1.8667 \\
 & & 0.1 & 2.9500 & 0.1 & 2.9500 \\
\hline
web-NotreDame & 0.2953 & 10 & 0.6261 & 10 & 0.6261 \\
 & & 1 & 1.4910 & 1 & 1.4910 \\
 & & 0.1 & 2.7938 & 0.1 & 2.7938 \\
\hline
citeulike-ut & 0.9509 & 10 & 0.0618 & 10 & 0.0618 \\
 & & 1 & 0.6948 & 1 & 0.6948 \\
 & & 0.1 & 2.9328 & 0.1 & 2.9328 \\
\hline
epinions & 0.2361 & 10 & 0.0355 & 10 & 0.0355 \\
 & & 1 & 1.6612 & 1 & 1.6612 \\
 & & 0.1 & 2.3498 & 0.1 & 2.3498 \\
\hline
epinions & 0.0098 & 10 & 0.0240 & 10 & 0.0240 \\
 & & 1 & 0.7403 & 1 & 0.7403 \\
 & & 0.1 & 2.7282 & 0.1 & 2.7282 \\
\hline
\end{tabular}
\caption{Relative error of our randomized average discriminative path length estimation algorithm.}
\end{table}

\begin{table}[h]
\centering
\begin{tabular}{|c|c|c|c|c|}
\hline
Database & Exact algorithm & Approximate algorithm & Sample size (%) & Relative error (%) \\
\hline
ADE (×1000) & 0.0183 & 10 & 0.0013 & 10 & 0.0013 \\
 & & 1 & 0.0013 & 0.1 & 0.0013 \\
\hline
dblp0305 & 0.0148 & 10 & 0.0007 & 10 & 0.0007 \\
 & & 1 & 0.0007 & 0.1 & 0.0007 \\
\hline
dblp0507 & 0.0154 & 10 & 0.0015 & 10 & 0.0015 \\
 & & 1 & 0.0015 & 0.1 & 0.0015 \\
\hline
dblp9202 & 0.0148 & 10 & 0.0096 & 10 & 0.0096 \\
 & & 1 & 0.0096 & 0.1 & 0.0096 \\
\hline
facebook-uniform & 0.0323 & 10 & 0.2627 & 10 & 0.2627 \\
 & & 1 & 1.1411 & 1 & 1.1411 \\
 & & 0.1 & 1.6568 & 0.1 & 1.6568 \\
\hline
citeulike-ut & 0.0898 & 10 & 0.2327 & 10 & 0.2327 \\
 & & 1 & 0.4596 & 1 & 0.4596 \\
 & & 0.1 & 4.4685 & 0.1 & 4.4685 \\
\hline
epinions & 0.0293 & 10 & 0.0749 & 10 & 0.0749 \\
 & & 1 & 0.04465 & 1 & 0.04465 \\
 & & 0.1 & 2.3459 & 0.1 & 2.3459 \\
\hline
epinions & 0.0450 & 10 & 0.0604 & 10 & 0.0604 \\
 & & 1 & 0.7933 & 1 & 0.7933 \\
 & & 0.1 & 0.8762 & 0.1 & 0.8762 \\
\hline
\end{tabular}
\caption{Relative error of our randomized average discriminative eccentricity estimation algorithm.}
\end{table}

\(^1\)Both the largest distance and the largest discriminative distance that two vertices may have in a graph are equal to the number of vertices in the graph minus 1.
bounded by a constant (i.e., 2). This also implies that average discriminative path length of a network is usually considerably smaller than its average path length.

This property means in real-world networks not only most vertices can be reached from every other vertex by a small number of steps, but also there are many different ways to do so. We call this property the tiny-world property. A consequence of this property is that removing several vertices from a real-world network, does not have a considerable effect on its average path length. In Table 4, it can be seen that networks such as flickr, petster-friendships, picstut and citeulike-ut have an average discriminative path length considerably smaller than the others. This is due to the high density of these networks, which yields that any two vertices may have a shorter distance or more shortest paths.

8 Link prediction in temporal graphs

In order to better motivate the applicability and usefulness of our proposed distance measure, in this section we present a novel link prediction method, which is based on our new notion. We empirically evaluate our method and show that it outperforms the well-known existing link prediction methods.

In the link prediction problem studied in this paper, we are given an unweighted and undirected graph $G$ in which each edge $e = \{u, v\}$ has a timestamp. For a time $t$, let $G[t]$ denote the subgraph of $G$ consisting of all edges with a timestamp less than or equal to $t$. Then the link prediction task is defined as follows. Given network $G[t]$ and a time $t'$, (partially) sort the list of all pairs of vertices that are not connected in $G[t]$, according to their probability (likelihood) of being connected during the interval $(t, t']$. We refer to the intervals $[0, t]$ and $(t, t']$ as the training interval and the test interval, respectively.

To generate this (decreasingly) sorted list, existing methods during the training interval compute a similarity matrix $S$ whose entry $S_{uv}$ is the score (probability/likelihood) of having an edge between vertices $u$ and $v$. Generally, $S$ is symmetric, i.e., $S_{uv} = S_{vu}$. The pairs of the vertices at the top of the ordered list are most likely to be connected during the test interval [42]. To compute $S_{uv}$, several methods have been proposed in the literature, including the number of common neighbors [45], negative of shortest path length [37] and its variations [33], the Jaccard’s coefficient [58], the preferential attachment index [4], hitting time [37], SimRank [28], Katz index [30], the Adamic/Adar index [1] and resource allocation based on common neighbor interactions [66]. In the literature, there are also many algorithms that exploit a classification algorithm, with these indices as the features, and try to predict whether a pair of unconnected vertices will be connected during the test interval or not [25, 39, 42].

In this section, we propose a new method for sorting the list of pairs of unconnected vertices, which is a combination of shortest path length and discriminative distance. The used ordering is called $\succ_d$ and is defined as follows. For two pairs of unconnected vertices $\{u_1, v_1\}$ and $\{u_2, v_2\}$, we say $\{u_1, v_1\} \succ_d \{u_2, v_2\}$ if:

- $d(u_1, v_1) > d(u_2, v_2)$, or
- $d(u_1, v_1) = d(u_2, v_2)$ and $dd(u_1, v_1) > dd(u_2, v_2)$.

Then, if $\{u_1, v_1\} \succ_d \{u_2, v_2\}$, vertices $u_2$ and $v_2$ are more likely to form a link during the test interval than vertices $u_1$ and $v_1$. The rationale behind $\succ_d$ is that when comparing a pair of vertices $u_1, v_1$ with another pair $u_2, v_2$, if $d(u_1, v_1) = d(u_2, v_2)$ but $u_2$ and $v_2$ are connected to each other by more shortest paths than $u_1$ and $v_1$, then they are more likely to form a link during the test interval. As a special case, for a fixed $k$, consider the list $L(k)$ consisting of all pairs of unconnected vertices $u$ and $v$ such that $d(u, v) = k$. A network may have many such pairs. It is known that compared to the pairs of unconnected vertices that have distance $k + 1$, members of $L(k)$ are more likely to form a link during the test interval [14, 51]. However, the question remaining open is what elements of $L(k)$ are more likely to be connected than the other members? Using $\succ_d$, we argue that by increasing the number of shortest paths between the two vertices, the probability of forming a link increases, too.

In order to empirically evaluate this argument, we perform tests over several temporal real-world networks, including sx-stackoverflow [50], sx-mathoverflow [50], sx-supersuser [50], sx-askubuntu [50], wiki-talk-temporal [34, 50] and CollegeMsg [49]. Table 6 summarizes the specifications of the used temporal real-world datasets. We consider all these networks as undirected graphs. Since the networks sx-stackoverflow, sx-supersuser, sx-askubuntu and wiki-talk-temporal are too large to load their unconnected pairs of vertices in the memory, after sorting their edges based on timestamp, we only consider the subgraphs generated by their first $300,000$ edges.

We compare $\succ_d$ with negative of shortest path length [37] and the Adamic/Adar index [1], denoted respectively by $\succ_\rho$ and $\succ_a$. We choose $\succ_\rho$ because $\succ_d$ is inherently an improvement of $\succ_\rho$. Furthermore, the experiments reported in [25] show that this index outperforms the other topological (global) indices studied in that paper. We choose $\succ_a$ because the experiments reported in [37] show that among 11 studied indices, the Adamic/Adar index has the best relative performance ratio versus random predictions, the best relative performance ratio versus negative of shortest path length predictor and the best relative performance ratio versus common neighbors predictor.

For the graph formed during training interval, we sort (increasingly when $\succ_d$ is used and decreasingly when negative of shortest path length and Adamic/Adar index are used) the list $L$ of all pairs of unconnected vertices, based on each of the indices$^3$. Then, during test interval, for each edge that connects a pair in $L$, we examine its rank in $L$. We define the ranking error of an index ind as follows.

$$Q(\text{ind}) = \sum_{\{u, v\} \in TE} \frac{\text{rank}((u, v), L_{\text{ind}})}{|TE|},$$

where $L_{\text{ind}}$ is the list $L$ sorted according to $\text{ind}$, $TE$ contains those edges of test interval that connect a pair in $L$, and $\text{rank}((u, v), L_{\text{ind}})$ returns the rank of $\{u, v\}$ in $L_{\text{ind}}$. For two given indices ind1 and ind2, $Q(\text{ind1}) < Q(\text{ind2})$ means that compared to $\text{ind1}$, $\text{ind2}$ gives more priority (i.e., a better rank) to the pairs that form a link during the test interval.

$^3$ When any of these indices is used, there might exist two or more pairs that are not sorted by the index. In this case, these pairs are sorted according to the identifiers of the end-points of the edges.
### Table 6: Specifications of the temporal real-world datasets used in our experiments for link prediction.

| Dataset            | Link                                                        | # vertices | # temporal edges | Time span   |
|--------------------|-------------------------------------------------------------|------------|------------------|-------------|
| sx-stackoverflow   | https://snap.stanford.edu/data/sx-stackoverflow.html        | 2,601,977  | 63,497,050       | 2774 days   |
| sx-mathoverflow    | http://snap.stanford.edu/data/sx-mathoverflow.html          | 24,818     | 506,550          | 2350 days   |
| sx-superuser       | https://snap.stanford.edu/data/sx-superuser.html             | 194,085    | 1,443,339        | 2773 days   |
| sx-askubuntu       | http://snap.stanford.edu/data/sx-askubuntu.html             | 159,316    | 964,437          | 2613 days   |
| wiki-talk-temporal | https://snap.stanford.edu/data/wiki-talk-temporal.html       | 1,140,149  | 7,833,140        | 2320 days   |
| CollegeMsg         | http://snap.stanford.edu/data/CollegeMsg.html                | 1,899      | 20,296           | 193 days    |

### Table 7: Empirical results of the link prediction algorithm. Ratio shows the percentage of the edges that form the training interval.

| Dataset       | Ratio | $Q(<a_{dd})$ | $Q(<a_d)$ | $Q(<a_{\bar{d}})$ |
|---------------|-------|--------------|-----------|-------------------|
| sx-stackoverflow | 60%   | 8841.43      | 12521.1   | 15230.6           |
|               | 70%   | 6638.36      | 9757.49   | 11753.9           |
|               | 80%   | 4524.19      | 6774.3    | 8107.91           |
|               | 90%   | 2360.62      | 3753.77   | 4514.21           |
| sx-mathoverflow | 60%   | 2415.41      | 4786.38   | 3916.04           |
|               | 70%   | 2436.96      | 4628.91   | 4243.17           |
|               | 80%   | 1695.37      | 3298.5   | 3128.66           |
|               | 90%   | 941.864      | 1684.61   | 1599.92           |
| sx-superuser   | 60%   | 4013.4       | 5733.22   | 7219.96           |
|               | 70%   | 2918.21      | 4261.63   | 5405.42           |
|               | 80%   | 2182.74      | 3052.82   | 3842.32           |
|               | 90%   | 1107.07      | 1646.94   | 2157.76           |
| sx-askubuntu   | 60%   | 2370.54      | 3255.43   | 4701.1            |
|               | 70%   | 2132.24      | 2855.66   | 4047.01           |
|               | 80%   | 1606.5       | 2167.51   | 3175.48           |
|               | 90%   | 881.154      | 1277.53   | 2115.77           |
| wiki-talk-temporal | 60%   | 3724.17      | 6728.74   | 8393.59           |
|               | 70%   | 3571.89      | 5826.38   | 7259.96           |
|               | 80%   | 1843.41      | 3394.38   | 4606.07           |
|               | 90%   | 783.075      | 1482.59   | 2029.89           |
| CollegeMsg     | 60%   | 1965.51      | 2501.29   | 2749.98           |
|               | 70%   | 1389.81      | 1757      | 1944.36           |
|               | 80%   | 1318.51      | 1820.25   | 2022.39           |
|               | 90%   | 558.038      | 890.692   | 995.216           |

Table 7 reports the empirical results. We sort the edges of each network according to their timestamps and form the training and test intervals based on the timestamps, i.e., for some given value $r$, training interval contains those edges that have a timestamp at most $r$ and test interval contains those edges that have a timestamp larger than $r$. A factor that may affect the empirical behavior of the indices is the value of $r$. Therefore and to examine this, we consider 4 different settings for each network, and choose the values of $r$ in such a way that training interval includes 60%, 70%, 80% and 90% of the edges. In each case, the rest of the edges belong to test interval. As can be seen in the table, in all the cases, $>_{dd}$ has the lowest ranking error and hence, the best performance. This empirically verifies our above mentioned argument that among all the pairs of unconnected vertices in $L(k)$, those that have a smaller discriminative distance (and hence, are closer!), are more likely to form a link. While in most cases $>_{dd}$ outperforms $>_d$, over sx-mathoverflow and for all values of ratio, $>_d$ has a lower ranking error than $>_{dd}$.

### 9 CONCLUSION AND FUTURE WORK

In this paper, we proposed a new distance measure between vertices of a graph, which is proportional to the length of shortest paths and inversely proportional to the number of shortest paths. We presented algorithms for exact computation of the proposed discriminative indices. We then developed effective randomized algorithms that precisely estimate average discriminative path length and average discriminative graph eccentricity and showed that they give $(\epsilon, \delta)$-approximations of these indices ($\epsilon \in \mathbb{R}^+$ and $\delta \in (0, 1)$). Then, we performed extensive experiments over several real-world networks from different domains. In our experiments, we first showed that compared to the traditional indices, discriminative indices have usually much more discriminability. We then showed that our randomized algorithms can very precisely estimate average discriminative path length and average discriminative eccentricity, using only a few samples. In the end, we presented a novel link prediction method that uses discriminative distance to decide which vertices are more likely to form a link in future, and showed its superior performance compared to the well-known existing measures.

The current work can be extended in several directions. An interesting direction is to investigate distribution of discriminative closeness and discriminative vertex eccentricity in large networks. In particular, it is useful to see whether there exist correlations among discriminative indices on the one hand and other centrality indices such as betweenness and degree on the other hand. The other direction for future work is to develop efficient randomized algorithms for estimating discriminative closeness and discriminative eccentricity of one vertex or a set of vertices and discriminative diameter of the graph. For example, it is interesting to develop algorithms similar to [6] that estimate $k$ highest discriminative closeness scores in the graph. Another extension of the current work is the empirical evaluation of the generalizations of the discriminative indices presented in Section 4.

### REFERENCES

1. Lada A. Adamic and Eytan Adar. 2001. Friends and Neighbors on the Web. *Social Networks* 25 (2001), 211–230.
2. R. Albert, H. Jeong, and A. L. Barabasi. 1999. The diameter of the world wide web. *Nature* 401 (1999), 130–131.
3. Frank Ball, Denis Mollison, and Gianpaolo Scalia-Tomba. 1997. Epidemics with two levels of mixing. *The Annals of Applied Probability* 7, 1 (1997), 46–89.
4. A. J. Barabasi, H. Jeong, Z. Neda, E. Ravasz, A. Schubert, and T. Vicsek. 2002. Evolution of the social network of scientific collaborations. *Physica A: Statistical
Mechanisms and its Applications, 311, 3 (2002), 590–614.

[5] David C. Bell, John S. Atkinson, and Jerry W. Carlson. 1999. Centrality measures for disease transmission networks. Soc. Networks 21, 1 (1999), 1–21.

[6] Elisabetta Bergamini, Michele Borassi, Pierluigi Crescenzi, Andrea Marino, and Henning Meyerhenke. 2016. Computing Top-k Closeness Centrality Faster in Unweighted Graphs. In Proceedings of the Eighteenth Workshop on Algorithm Engineering and Experiments,ALENEX 2016, Arlington, Virginia, USA, January 10, 2016. 68–80.

[7] Shiri Chechik, Daniel H. Larkin, Liam Roditty, Grant Schoenebeck, Robert E. Tarjan, and Virginia Vassilevska Williams. 2014. Better Approximation Algorithms for the Graph Diameter. In Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA ’14). Society for Industrial and Applied Mathematics, Philadelphia, PA, 1041–1052. http://dl.acm.org/citation.cfm?id=2634074.2634152

[8] Mostafa Haghir Chehreghani. 2014. An Efficient Algorithm for Approximate Betweenness Centrality Computation. Comput. J. 57, 9 (2014), 1371–1382. DOI: http://dx.doi.org/10.1093/comjnl/bxt003

[9] Mostafa Haghir Chehreghani and Mostafa Haghir Chehreghani. 2016. Modeling Transitivity in Complex Networks. In Proceedings of the Thirty-Second Conference on Uncertainty in Artificial Intelligence, UAI 2016, June 25-29, 2016, New York City, NY, USA, Alexander T. Ihler and Dimitris Fotakis (Eds.). AUAI Press. http://auai.org/uai2016/proceedings/papers/268.pdf

[10] Edith Cohen. 2000. Polylog-time and Near-linear Work Approximation Scheme for Undirected Shortest Paths. J. ACM 47, 1 (Jan. 2000), 132–166.

[11] Edith Cohen, Daniel Delling, Thomas Pajor, and Renato F. Werneck. 2014. Compressing Graphs Using Dynamic Programming on Trees. In Approximation and Online Algorithms, Part C: Emerging Technologies.

[12] Glen Jeh and Jennifer Widom. 2002. SimRank: A Measure of Structural-context Similarity. In Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD). ACM, 538–543.

[13] Mostafa Haghir Chehreghani, Albert Bifet, and Tarek Abdessalem. 2017. Link Prediction Using Top-k Shortest Distances. In Datasets Analysis - 31st British International Conference on Databases, BICOD 2017, London, UK, July 10-12, 2017, Proceedings (Lecture Notes in Computer Science), Andrea Calli, Peter T. Wood, Nigel J. Martin, and Alexandra Poulavassilis (Eds.), Vol. 10365. Springer, 101–105. DOI: http://dx.doi.org/10.1007/978-3-319-60795-5_10

[14] Andrei L Fitkevich, Jooyoung Lee, Victor Rivera, and Manuel Mazzara. 2017. Link Prediction Using Top-k Centrality Measures. In Social Networks: A Case Study of the Wikipedia Promotion Process. In Proc. Annual ACM-SIAM Symposium on Discrete Algorithms (SODA ’14). Society for Industrial and Applied Mathematics, Philadelphia, PA, 1041–1052. http://dl.acm.org/citation.cfm?id=2634074.2634152

[15] Jure Leskovec, Daniel Huttenlocher, and Jon Kleinberg. 2010. Governance in Social Media: A Case Study of the Wikipedia Promotion Process. In Proc. of 7th ACM International Conference on Web Search and Data Mining, WSDM 2010, New York, NY, USA, 423–432.

[16] Jure Leskovec, K. Lang, A. Dasgupta, and M. Mahoney. 2009. Community structure in large networks: Natural cluster sizes and the absence of large well-defined clusters. Internet Mathematics 6, 1 (2009), 29–123.

[17] David D. Lewis, Yiming Yang, Tony G. Rose, and Fan Li. 2004. RCV1: A New Benchmark Collection for Text Categorization Research. J. Machine Learning Research 5 (2004), 361–397.

[18] Zonghua Liu and Bambi Hu. 2005. Epidemic spreading in community networks. EPJ (Eulerplus Letters) 72, 2 (2003), 315.

[19] Linyuan Lu and Tao Zhou. 2011. A Survey of Link Prediction in Complex Networks: A Survey. Physica A: Statistical Mechanics and its Applications 390, 6 (2011), 1150–1170.

[20] Damien Magoni and Jean Jacques Pansiot. 2001. Analysis of the Autonomous System Network Topology. SIGCOMM Comput. Commun. Rev. 31, 3 (July 2001), 26–37.

[21] Paul W. Olsen, Alan G. Labouseur, and Jeong-Hyon Hwang. 2014. Efficient Top-k Closeness Centrality Search. In Proceedings of the Second ACM Conference on Online Social Networks (COSN ’14). ACM, New York, NY, USA, 37–50. DOI: http://dx.doi.org/10.1145/2660460.26604645

[22] Benjamin Cornwell. 2005. A complement-derived centrality index for disconnected graphs. Connections 28, 2 (2005), 708.

[23] Henning Meyerhenke. 2016. Computing Top-k Shortest Distances. In Proceedings of the Eighteenth Workshop on Algorithm Engineering and Experiments, ALENEX 2016, Arlington, Virginia, USA, January 10, 2016. 68–80.

[24] Mostafa Haghir Chehreghani. 2014. Effective Co-betweenness Centrality Computation. In Approximation and Online Algorithms, Part C: Emerging Technologies.

[25] Mohammad Al Hasan, Vineet Chaoji, Saeed Salem, and Mohammed Zaki. 2006. Link prediction using supervised learning. In Proceedings of the Second ACM International Conference on Web Search and Data Mining, WSDM ’06, New York, NY, USA, 423–432. DOI: http://dx.doi.org/10.1145/1151446.1151513

[26] Wassily Hoeffding. 1963. Probability Inequalities for Sums of Bounded Random Variables. J. Amer. Statist. Assoc. 58, 301 (March 1963), 13–30.

[27] M. Gjoka, M. Kurant, C. Butts, and A. Markopoulou. 2010. Walking in Facebook: A New Metric for Ranking Sociality. In Proceedings of the Thirteenth International Conference on Knowledge Discovery and Data Mining (KDD). ACM, 538–543.
Discriminative Distance-Based Network Indices, Link Prediction and the Tiny-World Property
Conference’17, July 2017, Washington, DC, USA

[51] M. Q. Pasta, F. Zaidi, and C. Rozenblat. 2014. Generating Online Social Networks based on Socio-demographic Attributes. Journal of Complex Networks 2, 4 (2014), 475–494.

[52] Georgios A. Pavlopoulos, Maria Secrier, Charalampos N. Moschopoulos, Theodoros G. Soldatos, Sophia Kossida, Jan Aerts, Reinhard Schneider, and Pan telis G. Bagos. 2011. Using graph theory to analyze biological networks. BioData Mining 4, 1 (2011), 10. DOI: https://dx.doi.org/10.1186/1756-0381-4-10

[53] Lorenzo Pellis, Neil M. Ferguson, and Christophe Fraser. 2011. Epidemic growth rate and household reproduction number in communities of households, schools and workplaces. Journal of Mathematical Biology 63, 4 (2011), 691–734.

[54] Christine S. Swart Peter Dankelmann, Wayne Goddard. 2004. The average eccentricity of a graph and its subgraphs. Utilitas Mathematica 65 (May 2004), 41–51.

[55] Abishai Polus. 1979. A study of travel time and reliability on arterial routes. Transportation 8, 2 (1979), 141–151.

[56] Yannick Rochat. 2009. Closeness centrality extended to unconnected graphs: The harmonic centrality index. In Applications of Social Network Analysis, ASNA.

[57] Liam Roditty and Virginia Vassilevska Williams. 2013. Fast Approximation Algorithms for the Diameter and Radius of Sparse Graphs. In Proceedings of the Forty-fifth Annual ACM Symposium on Theory of Computing (STOC ’13). ACM, New York, NY, USA, 515–524. DOI: http://dx.doi.org/10.1145/2488608.2488673

[58] Liam Roditty and Virginia Vassilevska Williams. 2013. Fast Approximation Algorithms for the Diameter and Radius of Sparse Graphs. In Proceedings of the Forty-fifth Annual ACM Symposium on Theory of Computing (STOC ’13). ACM, New York, NY, USA, 515–524. DOI: http://dx.doi.org/10.1145/2488608.2488673

[59] Frank W. Takes and Walter A. Kosters. 2013. Computing the Eccentricity Distribution of Large Graphs. Algorithms 6, 1 (2013), 100–118.

[60] Mateusz Krzysztof Tarowski, Piotr L. Szczepanski, Talal Rahwan, Tomasz P. Michalak, and Michael Wooldridge. 2016. Closeness Centrality for Networks with Overlapping Community Structure. In Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA., Dale Schuurmans and Michael P. Wellman (Eds.). AAAI Press, 622–629. http://www.aaaai.org/ocs/index.php/AAAI/AAAI16/paper/view/12149

[61] Jeffrey D. Ullman and Mihalis Yannakakis. 1991. High Probability Parallel Transitive-closure Algorithms. SIAM J. Comput. 20, 1 (Feb. 1991), 100–125. DOI: http://dx.doi.org/10.1137/0220006

[62] D. J. Watts and S. H. Strogatz. 1998. Collective dynamics of ‘small-world’networks. Nature 393, 6684 (1998), 409–10.

[63] Virginia Vassilevska Williams. 2012. Multiplying matrices faster than coppersmith-winograd. In Proceedings of the 44th Symposium on Theory of Computing Conference, STOC 2012. New York, NY, USA, May 19 - 22, 2012, Howard J. Karloff and Toniann Pitassi (Eds.). ACM, 887–898. DOI: http://dx.doi.org/10.1145/2213977.2214056

[64] Jianpei Zhang, Yuan Zhang, Hailu Yang, and Jing Yang. 2014. A Link Prediction Algorithm Based on Socialized Semi-local Information. Journal of Computational Information Systems 10 (2014), 4459–4466.