On Scalable Computation of Graph Eccentricities

Wentao Li
AAII, FEIT, University of Technology Sydney, Australia
wentao.li@uts.edu.au

Miao Qiao
University of Auckland, New Zealand
miao.qiao@auckland.ac.nz

Lu Qin
AAII, FEIT, University of Technology Sydney, Australia
lu.qin@uts.edu.au

Lijun Chang
The University of Sydney, Australia
lijun.chang@sydney.edu.au

Ying Zhang
AAII, FEIT, University of Technology Sydney, Australia
ying.zhang@uts.edu.au

Xuemin Lin
The University of New South Wales, Australia
lxue@cse.unsw.edu.au

ABSTRACT

Given a graph, eccentricity measures the distance from each node to its farthest node. Eccentricity indicates the centrality of each node and collectively encodes fundamental graph properties: the radius and the diameter — the minimum and maximum eccentricity, respectively, over all the nodes in the graph. Computing the eccentricities for all the graph nodes, however, is challenging in theory: any approach shall either complete in quadratic time or introduce a $\geq \frac{1}{3}$ relative error under certain hypotheses. In practice, the state-of-the-art approach PLLECC in computing exact eccentricities relies heavily on a precomputed all-pair-shortest-distance index whose expensive construction refrains PLLECC from scaling up. This paper provides insights to enable scalable exact eccentricity computation that *does not rely on any index*. The proposed algorithm IFECC handles billion-scale graphs that no existing approach can process and achieves up to *two orders of magnitude speedup* over PLLECC. As a by-product, IFECC can be terminated at any time during execution to produce approximate eccentricities, which is empirically more stable and reliable than kBFS, the state-of-the-art algorithm for approximately computing eccentricities.

CCS CONCEPTS

• **Mathematics of computing → Graph algorithms**.

KEYWORDS

eccentricity, diameter, radius, graph, algorithm, approximate

ACM Reference Format:
Wentao Li, Miao Qiao, Lu Qin, Lijun Chang, Ying Zhang, and Xuemin Lin. 2022. On Scalable Computation of Graph Eccentricities. In *Proceedings of the 2022 International Conference on Management of Data (SIGMOD '22), June 12–17, 2022, Philadelphia, PA, USA.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3514221.3517874

SIGMOD '22, June 12–17, 2022, Philadelphia, PA, USA

ACM ISBN 978-1-4503-9249-5/22/06…$15.00
https://doi.org/10.1145/3514221.3517874

1 INTRODUCTION

Graph is widely used to model interconnected objects: a graph $G(V, E)$ uses node set V to represent the collection of objects and the edge set E the interconnections among the objects. To analyze the proliferated graphs such as social networks, biological networks and web graphs, the measure of eccentricity plays a vital role.

Denote by $dist(u, v)$ the distance between two nodes u and v in the graph. The eccentricity of a node v is the longest distance from v to the other nodes in the graph: $ecc(v) = \max_{u \in V} dist(v, u)$. Eccentricity not only reflects the centrality of each node – a node's small eccentricity denotes its central location in the graph – but also collectively encodes fundamental global properties of a graph: the radius of the graph is the minimum eccentricity over all the nodes while the diameter the maximum. Therefore, the eccentricities $\{ecc(v)|v \in V\}$ over all the nodes in the graph, called the **eccentricity distribution**, becomes an important descriptor of the graph. In fact, eccentricity distribution has been used in characterizing routing networks [23], describing functional roles of proteins in biological networks [27], verifying formal hardwares [24], and boosting an individual's performance in social networks [21].

To further illustrate the significance of the exact and scalable eccentricity distribution computation, we provide the following real applications.

- **Location Optimization.** Define the vertices with the minimum eccentricity values as network center [14]. Network center has a very important role in location analysis. In placing time-critical facilities such as hospitals or fire stations, network center that minimizes service delays [15] is preferred. In improving the service quality by reducing the maximum delay over the network, storage centers will be placed based on network center [11].
- **Prediction Optimization.** The eccentricity distribution is used as a critical feature for various prediction tasks. It has been used in predicting missing edges in biological networks [36], identifying churners for telecom networks [31], finding spam users in social networks [20], and identifying infection sources of harmful information such as rumors or diseases [5]. Moreover, the eccentricities of all vertices can create the topological index [37, 38], which is a single number that servers as a reliable prediction of the physical, chemical, and biological properties for chemical compounds [39].

Unfortunately, the computation of eccentricity distribution has a quadratic barrier that cannot be solved by approximation in theory. It has been proved [28] that any approach of computing the

eccentricity distribution shall either complete in quadratic time or introduce a $\geq \frac{1}{3}$ relative error under strong exponential hypotheses. This implies that the attempt of finding a sub-quadratic approximate algorithm with a reasonably small error bound[1] would be in vain. Driven by this negative result and the high demand of analyzing the emerging massive small-world networks, e.g., social networks and web graphs, whose small eccentricities are less tolerant to errors, this paper aims at scaling up the computation of exact eccentricity distribution.

The straightforward computation of the eccentricity distribution is to perform $|V|$ Breadth-First-Searches (BFS), one BFS sourced from w, for each node w in the graph, in determining $ecc(w)$. To reduce the number of BFSs in computing the eccentricity distribution, a line of research [13, 33, 34], called BFS-based methods, associates, for the eccentricity of each node v, a lower bound and an upper bound which initially form a trivial range of $[0, +\infty)$. After performing a BFS from a source node t, one can reduce the gap between the eccentricity bounds of v using triangle inequalities (Section 3). If the gap becomes 0 on v after the update, then the v-sourced BFS can be avoided. Heuristics [13, 33, 34] have been explored to decide a priority order of the source node t of the BFSs such that v-sourced BFSs can be avoided for more graph nodes v. In using these heuristics, it is unclear how many BFSs can be avoided.

Compared to BFS-based methods, the state-of-the-art approach PLLECC [19] optimizes the exact eccentricity distribution computation in an orthogonal direction. Instead of reducing the number of BFSs in computing eccentricity distribution, PLLECC aims at reducing the cost of computing $ecc(v)$ for each node v. Note that performing a BFS from v cannot determine $ecc(v)$ until all the nodes in V have been visited, due to its "from-near-to-far" (w.r.t. v) order in visiting nodes in the graph. PLLECC adopts a reverse-BFS order of v, called the Farthest-First Node Order (FFO), in visiting each node t in the graph. PLLECC probes $dist(t, v)$, for each node t, to update the eccentricity bounds of v until the gap of v is closed. The FFO tightens the upper and lower bounds of $ecc(v)$ faster: i) the farthest node t from v can update the lower bound to $dist(v, t) = ecc(v)$ and thus makes the lower bound tight and ii) the FFO derives an extra upper bound on the distances from unvisited nodes to v (Section 3).

The effectiveness of PLLECC comes with a strong dependency on a precomputed all-pair-shortest-distance index whose construction becomes the bottleneck of PLLECC on massive graphs. The dependency is hard to remove. Specifically, seeing that computing the FFOs for all the graph nodes takes quadratic time and is thus unaffordable, PLLECC creates an "approximate" FFO for each node v. These approximate FFOs are computed by selecting a set Z of **reference nodes** and then letting each node v borrow the FFO of its closest reference node in Z. This way, it suffices to compute and store the exact FFOs of nodes in Z. The approximate FFOs work since 1) near-by nodes tend to share similar FFOs and 2) in small-world networks, it is easy to select a small number of reference nodes that are close to most of the other nodes in the graph. However, to follow these approximate FFOs and then handle the scattered distance probing queries, PLLECC has to construct an all-pair-shortest-distance index which becomes the bottleneck on big graphs. For graph SK (see Table 3 for dataset's details) with 1.9 billion edges, the index size exceeds 190 GB; for graph IT with 1.2 billion edges, the index size goes beyond 400 GB.

Existing BFS-based methods lay their efficiency on a heuristically selected priority order while the state-of-the-art approach PLLECC depends on an expensive all-pair-shortest-distance index; none of them can compute the exact eccentricity distribution with reasonable spatial-temporal resources for the billion-scale graphs emerging nowadays. To solve this problem, we propose an approach IFECC with explainable superiority on computing the exact eccentricity distribution. Our contributions are summarized below.

(1) We abstract the general BFS-framework which captures various approaches for computing both exact and approximate eccentricity distributions. We analyze the state-of-the-art approach PLLECC in-depth to show the tight connection between its efficiency and the employed all-pair-shortest-distance index.
(2) We propose an approach IFECC which i) conforms to the BFS-framework and ii) non-trivially decouples PLLECC from the distance index and thus improves the efficiency and scalability.
(3) We introduce new parameters with statistical significance for analyzing the complexity of IFECC to achieve better theoretical results under the quadratic barrier of the exact computation of the eccentricity distribution. We are the first in conducting such an analysis which unveils the nature of the problem. We also provide statistics of the new parameters on large graphs.
(4) We verify the superiority of IFECC with extensive empirical results. IFECC is more than one order of magnitude faster than the state-of-the-art exact approach on average and more importantly, can process billion-scale graphs that cannot be processed by any existing approach.
(5) As a by-product, IFECC can be terminated at any time during execution to produce approximate eccentricities, which can provide a competitive approximation in comparison with the state-of-the-art approximate approach kBFS [32]. Specifically, under the same temporal-spatial resource, our approach provides a more stable and reliable eccentricity distribution estimation on real-world graphs.

Section 2 defines the problem. Section 3 introduces the BFS-framework and the state-of-the-art method for eccentricity computation. Section 4 proposes IFECC and Section 5 analyzes the efficiency of IFECC. Section 6 shows the related work. Section 7 exhibits the experimentations and Section 8 concludes the paper.

2 PROBLEM DEFINITION

This section formally defines the concepts that will be used in the paper. Some of the concepts may been mentioned in Section 1.

Let $G(V, E)$ be an unweighted and undirected graph with a set V of $n = |V|$ vertices and a set E of $m = |E|$ edges. An edge $e(u, v) \in E$ connects two vertices $u, v \in V$. For a vertex $v \in V$, its degree $deg(v)$ is the number of neighbors adjacent to v, i.e., $deg(v) = |\{u|e(u, v) \in E\}|$. A path $p(s, t)$ from a node s to a node t in the graph is a sequence of edges $e_1(s, v_1), e_2(v_1, v_2), \cdots, e_{l-1}(v_{l-2}, v_{l-1}), e_l(v_{l-1}, t) \in E$. The length of the path $|p(s, t)| = l$ is the number of edges on the path. The distance $dist(s, t)$ between s and t is the length of the shortest

[1]Note that an error bound is different from the empirical statistics on the error.

Table 1: Frequently Used Notations

Notation	Description
$G(V, E)$	graph G with vertices V and edges E
$dist(s, t)$	distance between s and t
$ecc(v)$	eccentricity of vertex $v \in V$
$\underline{ecc}(v)$, $\overline{ecc}(v)$	lower bound and upper bound of $ecc(v)$
L^z	FFO of the reference node z
$\mathsf{PN}^z(v_i^z)$	probe number of $v_i^z \in L^z$
r	number of reference nodes
k	sample size used for approximate algorithms

path from s to t. The distances from a node to all the other nodes can be computed in a Breadth-First-Search (BFS) whose complexity is $O(m + n)$.

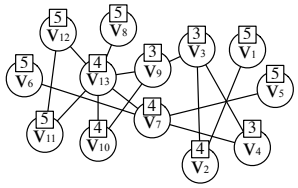

Figure 1: The Example Graph G

Example 2.1. Figure 1 presents graph G of the running example. It has 13 nodes and 15 edges. $deg(v_{10}) = 2$ and $dist(v_{10}, v_{12}) = 2$.

Definition 2.2. Given a vertex v of a graph $G(V, E)$, the **eccentricity** of v is $ecc(v) = \max_{u \in V} dist(v, u)$. The **farthest node** to v is $f_v = argmax_{u \in V} dist(v, u)$. If the farthest node of v is not unique, f_v can be any of the farthest nodes. The **eccentricity distribution** ED of G is $\mathsf{ED}(G) = \{ecc(v) | v \in V\}$.

The eccentricity distribution can derive the radius rad and diameter dia in linear time, that is, $rad = \min_{v \in V} ecc(v)$ and $dia = \max_{v \in V} ecc(v)$.

Example 2.3. Figure 1 labels the eccentricity of each node in G. $ecc(v_{10}) = 4$ while the farthest node $f_{v10} = v_1$ since $dist(v_1, v_{10}) = 4$. The radius rad is 3 and the diameter dia is 5.

We assume that G is a connected graph in the paper while the results can be extended to disconnected graphs easily[2]. For the sake of clarity, we summarize the commonly used symbols in Table 1.

3 PROBLEM ANALYSIS

This section first abstracts BFS-framework from recent studies [2, 10, 12, 13, 28, 32–34] on exact and approximate computations of the eccentricity distribution and then introduces the state-of-the-art approach for computing exact eccentricity distribution.

3.1 BFS-Framework

The eccentricity distribution can be computed by performing $|V|$ BFSs, one BFS from each node t to determine $ecc(t)$. To reduce the number of BFSs, a common practice is to attach a lower bound $\underline{ecc}(v)$ and an upper bound $\overline{ecc}(v)$ to each node $v \in V$. When the eccentricity of a node t is obtained by performing a t-sourced BFS, other nodes can update their bounds using Lemma 3.1.

Lemma 3.1 (Bound Update [34]). *Let t be a node of graph $G(V, E)$ with eccentricity $ecc(t)$. Given a node v and its distance $dist(v, t)$,*

$$ecc(v) \leq dist(v, t) + ecc(t) \quad (1)$$

$$ecc(v) \geq \max\{dist(v, t), ecc(t) - dist(v, t)\}. \quad (2)$$

Proof. See Appendix. □

Upon computing $ecc(t)$, one can update the upper bound $\overline{ecc}(v)$ of v with $dist(v, t) + ecc(t)$, the lower bound $\underline{ecc}(v)$ with $\max\{dist(v, t), ecc(t) - dist(v, t)\}$; if one only knows $dist(v, t)$, he can also update $\underline{ecc}(v)$. When the lower bound on v meets the upper bound, $ecc(v)$ is determined without performing a v-sourced BFS.

BFS-framework. We abstract the BFS-framework which captures existing approaches [2, 10, 12, 13, 28, 32–34] in computing the eccentricity distribution either exactly or approximately.

(1) Initialize the eccentricity upper bound $\overline{ecc}(v) = +\infty$ and lower bound $\underline{ecc}(v) = 0$ for each node v in the graph;
(2) Determine the source node set $S \subseteq V$. S is selected either collectively or gradually from an initially empty set along a predefined priority order;
(3) For each node t in S, compute the BFS from t, and update the eccentricity bounds of other nodes v using Lemma 3.1.

Limitation. The performance of the methods under the BFS-framework is heavily impacted [19] by the priority order/node selection strategy of S in Step 2; none of these heuristics (e.g., [33, 34]) can provide an analysis on the number of BFSs needed. It remains open to set a desirable priority order/node selection strategy of S.

3.2 The State-of-the-Art Exact ED Computation

Compared to the approaches under the BFS-framework, the state-of-the-art approach PLLECC [19] optimizes the exact computation of the eccentricity distribution in an orthogonal direction. Instead of reducing the number of BFSs, PLLECC aims at determining $ecc(v)$ with less effort than a linear search of BFS, for each node $v \in V$.

Farthest-First Node Order (FFO). PLLECC notices that the complexity of computing $ecc(v)$ cannot be improved using v-sourced BFS: the BFS from v visits nodes in a near-to-far order and thus $ecc(v)$ cannot be determined until all the nodes in V have been visited. To enable an early stop, PLLECC traverses nodes $t \in V$ following a reverse-BFS (of v) order, called the Farthest-First Node Order (FFO). It probes distance $dist(t, v)$ to update the eccentricity bounds of v until the gap of v becomes 0. Note that the quadratic time of computing the FFO for each node v is unaffordable, PLLECC thus adopts, for each node v, an approximate FFO generated below.

(1) Select a few high-degree nodes to form a set Z of **reference nodes** — high-degree nodes are usually located at the center of a graph and near other nodes [9, 19].
(2) Compute, for each node z in Z, z's FFO $L^z = \langle v_1^z, v_2^z, \cdots, v_n^z = z \rangle$ such that $dist(z, v_1^z) \geq dist(z, v_2^z) \geq \cdots \geq dist(z, v_n^z)$.
(3) For node $v \in V$, let **the reference node z of v** be the node in Z closest to v. Let the approximate FFO of v be the FFO of z.

[2]If one defines the eccentricity as $+\infty$ when the graph is disconnected, computing the eccentricity distribution becomes trivial; otherwise, one can process each connected component of the graph respectively with our solution.

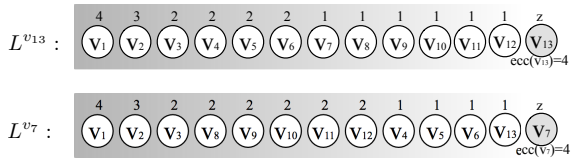

Figure 2: Farthest-First Node Order

Example 3.2. For the graph in Figure 1, the reference nodes Z include two nodes v_{13}, v_7 with the highest degrees. For each node in Z, the farthest-first node order is given in Figure 2. For example, $L^{v_{13}} = \{v_1, v_2, v_3, \cdots, v_{13}\}$ — all nodes are arranged in a non-increasing order of their distances to v_{13}.

Early-stop in Computing the Eccentricity of a Node. For a node v, PLLECC probes the distances based on the FFO of v's reference node z, which enables an early-stop in computing $ecc(v)$. Specifically, the distance from nodes in L^z to v is sequentially probed while the upper bound $\overline{ecc}(v)$ and lower bound $\underline{ecc}(v)$ of v updated along the probing. The lower bound can be updated by Lemma 3.1 and the upper bound by Lemma 3.3. The probing stops when $\overline{ecc}(v) = \underline{ecc}(v)$.

Lemma 3.3 (Upper Bound [19]). *Given v and its reference node z, consider the moment when the distances from all nodes in $T = \{v_1^z, v_2^z, \cdots, v_i^z\} \subseteq L^z$ have been probed.*

$$ecc(v) \leq \max\{\underline{ecc}(v), dist(v_i^z, z) + dist(z, v)\}, \tag{3}$$

where $\underline{ecc}(v) = \max_{u \in T} dist(u, v)$.

Proof. See Appendix. □

In Equation 3, term $\underline{ecc}(v)$ never decreases along the probing but never exceeds $ecc(v)$, and term $dist(v, z) + dist(v_i^z, z)$ never increases along the probing under FFO L^z. Thus, the gap between the lower and upper bounds can decrease in a fast pace along the probing. Therefore, the effectiveness in closing the gap between the eccentricity bounds in PLLECC heavily relies on the distance probing under FFO. Since the benefit of the early-stop can be easily cancelled out if the distance of $dist(v, v_i^z)$ cannot be efficiently probed, a precomputed all-pair-shortest-distance index for pair-wise distance queries is essential to the efficiency of PLLECC. PLLECC adopts the cutting-edge method PLL [3] to create the distance index [19].

Algorithm. Algorithm 1 shows the two stages of PLLECC. In the first stage, denoted as PLLECC-PLL, the distance index is created with PLL (Line 1). In the second stage, denoted as PLLECC-ECC, PLLECC initializes a small set of r (r is a parameter) reference nodes $Z \subseteq V$ (Line 2), then computes, for each node z in Z, $ecc(z)$ and the FFO $L^z = \langle v_1^z, v_2^z, \cdots, v_n^z \rangle$ of z by performing a z-sourced BFS (Line 4-5). After that, in computing the eccentricity of a node v (Line 6-14), PLLECC picks the node z in Z that is closest to v (Line 7). The upper bound and lower bound of v are updated by Lemma 3.1 (Line 8-9). Then, nodes in L^z are probed sequentially to update the eccentricity bounds of v (Line 10). Specifically, for each probed node v_i^z, the distance between v and v_i^z is obtained by exploiting the distance index (Line 11). The node order L^z (Line 4) derives a new upper bound (Line 12-13) of $ecc(v)$ (Lemma 3.3). The probe for v terminates when the bounds of v match (Line 14).

Algorithm 1: PLLECC

Input: Graph $G(V, E)$, reference number r
Output: Eccentricity distribution ED(G)
// PLLECC-PLL stage
1 Create the distance index by PLL;
// PLLECC-ECC stage
2 Select a small set Z of r high degree nodes;
3 **for** *each node* $z \in Z$ **do**
4 $\quad ecc(z) \leftarrow$ a BFS from z;
5 $\quad$ Generates $L^z = \langle v_1^z, v_2^z, \cdots, v_n^z \rangle$ for z;
6 **for** *each node* $v \in \{V \setminus Z\}$ **do**
7 $\quad z \leftarrow$ the node in Z that is closet to v;
$\quad$ // Apply Lemma 3.1
8 $\quad \underline{ecc}(v) \leftarrow \max\{\underline{ecc}(v), dist(v, z), ecc(z) - dist(v, z)\}$;
9 $\quad \overline{ecc}(v) \leftarrow \min\{\overline{ecc}(v), ecc(z) + dist(v, z)\}$;
10 $\quad$ **for** *sequentially each node* $v_i^z \in L^z$ **do**
11 $\quad\quad$ Compute $dist(v, v_i^z)$ from the index;
$\quad\quad$ // Apply Lemma 3.1 and 3.3
12 $\quad\quad \underline{ecc}(v) \leftarrow \max\{\underline{ecc}(v), dist(v, v_i^z)\}$;
13 $\quad\quad \overline{ecc}(v) \leftarrow$
$\quad\quad\quad \min\{\overline{ecc}(v), \max\{\underline{ecc}(v), dist(v_i^z, z) + dist(z, v)\}\}$;
14 $\quad\quad$ **if** $\underline{ecc}(v) = \overline{ecc}(v)$ **then break**;
15 **return** ED(G)

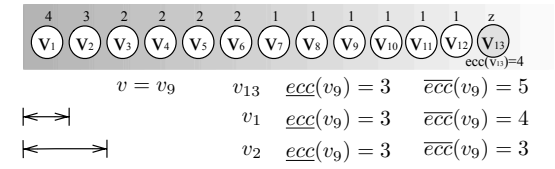

Figure 3: The Illustration of PLLECC

Example 3.4. Given two reference nodes $Z = \{v_{13}, v_7\}$ and their farthest-first node orders in Figure 2. Figure 3 demonstrates the computation of $ecc(v)$ with $v = v_9$. First, v_9 selects the closest node v_{13} from Z as the reference node z. Then, the lower bound of v_9 is updated to $ecc(z) - 1 = 3$ and the upper bound is updated to $ecc(z) + 1 = 5$ by Lemma 3.1. Then, v_9 probes the nodes in $L^{v_{13}}$ sequentially to update the bounds. When v_1 is visited, the lower bound remains 3 as $dist(v_1, v_9) = 3$ and the upper bound will be $dist(v_2, z) + dist(v_9, z) = 4$. Then, v_2 is visited to update the upper bound to $dist(v_3, z) + dist(v_9, z) = 3$. This determines $ecc(v_9) = 3$.

Limitation. The distance index is crucial for the efficiency of Algorithm 1 whose construction becomes the spatial-temporal bottleneck of PLLECC. As our experiments shall show, the construction of the index dominates the total time of PLLECC [19]. Moreover, for graph SK with 1.9 billion edges, the index size is larger than 190 GB; for graph IT with 1.2 billion edges, the index is > 400 GB.

4 INDEX-FREE ED COMPUTATION

This section proposes an index-free approach called IFECC that scales up the eccentricity distribution computation of PLLECC.

4.1 PLLECC Revisit

The distance index that encodes all-pair shortest distances can be an overkill for PLLECC. Recall that the index is only used to probe the distance $dist(v_i^z, v)$ when computing $ecc(v)$ for a node v. Here z is the reference node of v and $v_i^z \in L^z$. We now take an alternative perspective. Let z be a node in Z, for each node $v_i^z \in L^z$, we consider the nodes $v \in V$ whose i) reference node is z and ii) distance $dist(v, v_i^z)$ to v_i^z have been probed by PLLECC. The shift in perspective derives a new measure – the frequency that v_i^z is engaged in the distance probing – that reflects the importance of each node v_i^z in L^z.

*Definition 4.1 (**Probe Number**).* For any $z \in Z$, consider $v_i^z \in L^z$, define the probe number $\mathsf{PN}^z(v_i^z)$ as the total number of distance queries posed by PLLECC in the form of $dist(v, v_i^z)$ such that z is the reference node of v.

Table 2: The Probe Number

$L^{v_{13}}$	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}	v_{11}	v_{12}	v_{13}
$\mathsf{PN}^{v_{13}}$	7	3	1	0	0	0	0	0	0	0	0	0	0
L^{v_7}	v_1	v_2	v_3	v_8	v_9	v_{10}	v_{11}	v_{12}	v_4	v_5	v_6	v_{13}	v_7
PN^{v_7}	3	1	1	0	0	0	0	0	0	0	0	0	0

Example 4.2. In Figure 3, to determine the eccentricity of v_9, v_9 selects v_{13} as the reference node and then visits nodes sequentially in $L^{v_{13}}$. During this process, nodes $\{v_1, v_2\}$ have been probed by v_9, and we increase their probe number regarding v_{13} by one. When all nodes using v_{13} as the reference node find the eccentricity, we obtain the final probe number regarding v_{13} in Table 2.

Property of Probe Number. Each node v visits nodes in L^z from v_1^z to v_n^z sequentially, thus, $\mathsf{PN}^z(v_1^z)$ is the max among nodes in L^z — other nodes cannot be probed if v_1^z has not been visited. We generalize this observation and provide the relation between the probe number regarding z and the location of a node in L^z in Lemma 4.3.

Lemma 4.3. *For a reference node z and two nodes v_i^z and v_j^z in L^z, $\mathsf{PN}^z(v_i^z) > \mathsf{PN}^z(v_j^z)$ only if $i < j$.*

Proof. Suppose $\mathsf{PN}^z(v_i^z) > \mathsf{PN}^z(v_j^z)$, but $i \geq j$. The fact $i \geq j$ means any node $v \in V$ that visits v_i^z must have visited v_j^z, which implies $\mathsf{PN}^z(v_i^z) \leq \mathsf{PN}^z(v_j^z)$, contradiction. □

Lemma 4.3 indicates that nodes at the front of L^z of each $z \in Z$ have large probe numbers while the nodes at the tail of L^z might have not been probed under z at all. If we can remove the nodes whose probe number is 0 from the index, then the index size may be significantly reduced.

Example 4.4. In Table 2, for both v_{13} and v_7, we find that nodes with small subscripts in the farthest-first order L^z have probe numbers no smaller than nodes with large subscripts. Moreover, nodes with large subscripts are likely to have zero probe number.

Algorithm 2: IFECC

Input: Graph $G(V, E)$, reference number r
Output: Eccentricity distribution $\mathsf{ED}(G)$

1 Select a small set Z of r high degree nodes;
2 **for** *each node* $z \in Z$ **do**
3 $V^z \leftarrow \emptyset$;
4 Compute $ecc(z)$ and FFO $L^z = \langle v_1^z, v_2^z, \cdots, v_{n-1}^z \rangle$ of z by performing a BFS from z;
5 **for** *each node* $v \in \{V \setminus Z\}$ **do**
6 $z \leftarrow$ the node in Z that is closet to v;
7 Insert v to V^z;
  `// Apply Lemma 3.1`
8 $\underline{ecc}(v) \leftarrow \max\{\underline{ecc}(v), dist(v, z), ecc(z) - dist(v, z)\}$;
9 $\overline{ecc}(v) \leftarrow \min\{\overline{ecc}(v), ecc(z) + dist(v, z)\}$;
10 **for** *each node* $z \in Z$ **do**
11 $n_z \leftarrow |V^z|$;
12 **for** *each node* $v_i^z \in L^z$, *sequentially,* **do**
13 Perform BFS from v_i^z;
14 **for** *each node* $v \in V^z$ *with* $\underline{ecc}(v) \neq \overline{ecc}(v)$ **do**
15 $\underline{ecc}(v) \leftarrow \max\{\underline{ecc}(v), dist(v, v_i^z)\}$;
  `// Apply Lemma 3.1 and 3.3`
16 $\overline{ecc}(v) \leftarrow \min\{\overline{ecc}(v), \max\{\underline{ecc}(v), dist(v_i^z, z) + dist(z, v)\}\}$;
17 **if** $\underline{ecc}(v) = \overline{ecc}(v)$ **then** $n_z \leftarrow n_z - 1$;
18 **if** $n_z = 0$ **then break**;
19 **return** $\mathsf{ED}(G)$

4.2 Space-Efficient Computation

The index size can be reduced by removing the nodes whose probe number is zero under all $z \in Z$. However, how can we know whose probe number is zero without carrying out PLLECC? To answer this question, we revisit the BFS-framework (introduced in Section 3).

Farthest-First Node Order. Recall that the methods under BFS-framework struggle to find a priority order. Lemma 4.3 indicates that nodes at the beginning of L^z are important in PLLECC. In this sense, shall we set the priority order based on L^z?

Algorithm. We plug the farthest-first node order of z into the BFS-framework and obtain the index-free solution IFECC, see Algorithm 2, for computing the exact eccentricity distribution. Similar to PLLECC, IFECC selects r highest degree nodes Z as the reference nodes and generates the FFO L^z, for $\forall z \in Z$ (Line 1-4). Each node $v \in V$ finds its reference node and inserts itself into the territory V^z of z. The lower and upper bound of v is updated by z using Lemma 3.1 (Line 5-9). After that, each reference node z is responsible to decide the eccentricities for all the nodes in the territory V^z of z (Line 10-18). Specifically, we visit nodes v_i^z in L^z sequentially to obtain the distances from v_i^z to all nodes in V (Line 12-13). In this way, the bounds of each node $v \in V^z$ can be updated by Lemma 1-2, which is similar to Algorithm 1 since we replace the distance index probing with explicit distance computation of v_i^z. The loop for

$z \in Z$ ends when all the nodes in V^z have found their eccentricity (Line 17-18).

Theorem 4.5. *The space complexity of Algorithm 2 is $O(m+n)$.*

Proof. Algorithm 2 requires: i) $O(m+n)$ space to load the entire graph; ii) $O(n)$ space to store $ecc(v)$, $\underline{ecc}(v)$, $\overline{ecc}(v)$ for all vertices $v \in V$. Therefore, the total space cost is $O(m+n)$. □

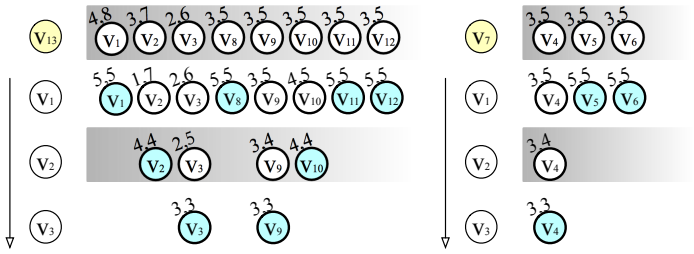

Figure 4: The Illustration of IFECC

Example 4.6. Figure 4 presents the process of IFECC on the graph in Figure 1. IFECC selects 2 reference nodes v_{13} and v_7 and divides nodes into two groups based on their reference nodes. $V^{v_{13}} = \{v_1, v_2, v_3, v_8, v_9, v_{10}, v_{11}, v_{12}\}$ finds v_{13} as the reference node and $V^{v_{17}} = \{v_4, v_5, v_6\}$. For v_{13}, $v_1 \in L^{13}$ conducts BFS to update the bounds of other vertices. This determines the eccentricities of $\{v_1, v_8, v_{11}, v_{12}\}$. This process continues when v_3 performs BFS and all vertices have exact eccentricities. The process for v_7 is the same. The number of BFSs performed for v_{13} is 4, for v_7 is 4, and 8 in total.

Remark. *In Algorithm 2, the distances from the reference nodes Z to the other nodes need to be stored (Line 2-4). In practice, we can make Z contain only one node, as described in Section 4.2. Therefore, these additional space requirements are negligible. Instead,* PLLECC *needs to store the distance index, which encodes the distances between all node pairs. We further compare the space consumption of our approach and* PLLECC *in Section 7.2.*

4.3 Time-Efficient Computation

This section shows an insight uniquely given by IFECC that cannot be observed from PLLECC: one reference node is sufficient for an efficient computation; the novel bounds derived under 1 reference node shall be elaborated in Section 5. This section ends with an approximation adaptation to further accelerate the computation.

Reference Node Number. The number of reference nodes, denoted by r, is a parameter of PLLECC. Choosing multiple reference nodes enables PLLECC to exploit Lemma 3.3 for an effective bound update [19]. By contrast, in IFECC, multiple reference nodes lead to redundant BFS computations: a node in the outskirt of the graph may simultaneously locate at the beginning of the FFOs of multiple reference nodes.

We examine two representative networks: web graph IT-2004 (denoted by IT) and social network Twitter (denoted by TWIT). Let the number of reference nodes be 16 — the default reference node number in PLLECC. Denote by D_z the set of the first *num* nodes in the FFO of each reference node $\forall z \in Z$ where *num* is an experiment

Algorithm 3: kIFECC

Input: Graph $G(V, E)$, sample size k
Output: Approximate Eccentricity Distribution $\widetilde{\mathrm{ED}}(G)$

1 Let Z include the highest-degree node z in the graph;
2 Line 2-9 of Algorithm 2;
3 **for** i from 1 to k and $v_i^z \in L^z$ **do** Line 13-16 of Algorithm 2;
4 **return** $\widetilde{\mathrm{ED}}(G) \leftarrow \{\underline{ecc}(v) | v \in V\}$

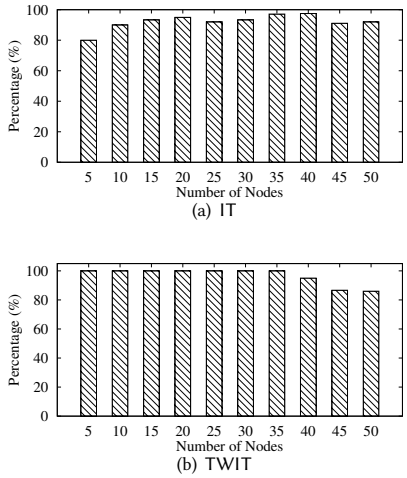

Figure 5: Number of Common Nodes

parameter. Figure 5 shows the repetition ratio $\frac{|\bigcap_{z\in Z} D_z|}{|\bigcup_{z\in Z} D_z|}$ when *num* takes 5, 10, $\cdots$, 50, respectively. On average, more than 94.5% of high-probe-number nodes are shared by all reference nodes.

To reduce the redundancy in computing the BFSs, one may memorize the computed results to avoid re-computation; however, this imposes additional space cost for IFECC.

One Reference Node is Enough. The redundancy can be thoroughly removed by setting the reference node number r to 1. Setting $r = 1$ not only works efficiently in practice (Section 7) but also sheds light on theoretical analysis which will be illustrated in Section 5.

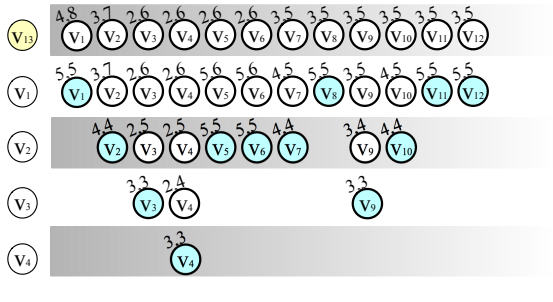

Figure 6: Setting One Reference Node for IFECC

Example 4.7. Figure 4 shows that 8 BFSs are needed when the reference node number is 2. Redundant BFSs are performed since

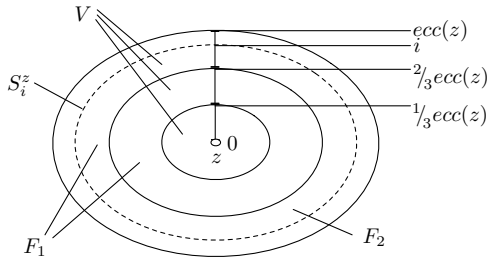

Figure 7: Stratified Graph from z

v_1, v_2, v_3 are the common distant nodes shared by v_{13} and v_7. Figure 6 presents IFECC under 1 reference node. The total number of BFSs is reduced to $4+1=5$, smaller than 8 under 2 reference nodes.

Approximation Adaptation. Using only one reference node, one can make a trade-off between the computation time and precision. Rather than waiting until the bounds match on all the nodes, we can control the time with the sample size k, a parameter of the number of BFSs to perform. Algorithm 3 adapts IFECC to an approximation algorithm kIFECC by terminating immediately after k nodes in L^z have conducted BFSs. The effectiveness of kIFECC will be theoretically analyzed in Section 5 and empirically evaluated in Section 7.

5 THEORETICAL ANALYSIS

This section introduces two important theorems in proving the effectiveness of IFECC with only one reference node z (i.e., $Z=\{z\}$). Under the BFS-framework, the performance of IFECC is reflected by the matching efficiency of the eccentricity bounds — the only way that one can improve the efficiency of the exact eccentricity computation due to the negative results in [28]. The result can be shared by both exact and approximate versions of IFECC.

Stratified Graph. To simplify the analysis, we first stratify the graph in Definition 5.1 (see Fig. 7). Our analysis holds for any reference node z, in other words, no property of node z is used.

Definition 5.1. Given a node $z \in V$, define the i-th **layer**, $S_i^z = \{v | dist(v,z) = i\}$, as the set of nodes whose distances to z equal to i.

Example 5.2. The nodes in V are partitioned according to their distances to $z = v_{13}$. $ecc(z) = 4$. Five layers of z: $S_0^z = \{v_{13}\}$, $S_1^z = \{v_7, v_8, v_9, v_{10}, v_{11}, v_{12}\}$; $S_2^z = \{v_3, v_4, v_5, v_6\}$; $S_3^z = \{v_2\}$; $S_4^z = \{v_1\}$.

The layer division of a graph G under the reference node z groups nodes according to their distances to z. We focus on the nodes who lie at least $\frac{1}{3}ecc(z)$ (and $\frac{2}{3}ecc(z)$, respectively) far away from z.

Definition 5.3. Tripartite the layers to define two sets:

Farthest (2/3) set: $F_1 = \{v \in V | dist(v,z) > (1/3)ecc(z)\}$,

Farthest (1/3) set: $F_2 = \{v \in V | dist(v,z) > (2/3)ecc(z)\}$.

For a set S and a node v, denote by $dist_{max}(v,S) = \max_{u \in S}\{dist(u,v)\}$ the **maximum set distance** from v to S.

Example 5.4. F_1 has the nodes in last $\frac{2}{3}ecc(z) = 3$ layers and thus $F_1 = \{v_1, v_2, v_3, v_4, v_5, v_6\}$ and $F_2 = S_3^z \cup S_4^z = \{v_1, v_2\}$.

Analysis of Exact Algorithm. Rather than obtaining an $O(nm)$ worse-case complexity as the general methods under the BFS-framework, we prove that the proposed IFECC derives a better complexity.

Theorem 5.5. *Performing BFSs from nodes in F_1 in $O(|F_1|(m+n))$ time computes the eccentricity distribution of the graph.*

Proof. The eccentricities of nodes in F_1 can be directly obtained. Next, we prove that for $\forall v \notin F_1$, $ecc(v) = \max_{u \in F_1}\{dist(u,v)\}$; in other words, for $\forall v \notin F_1$, there is a node $u \in F_1$ such that $dist(u,v) = ecc(v)$, by contradiction. Assume that none of the farthest nodes of v is in F_1. Let w be a farthest node of v. Since $w \notin F_1$, $dist(w,z) \le \frac{1}{3}ecc(z)$. $dist(v,w) \le dist(v,z) + dist(z,w) \le \frac{2}{3}ecc(z)$. Let f_z be the farthest node of z, that is, $ecc(z) = dist(z, f_z)$. We have $f_z \in F_1$. Besides, $dist(f_z, v) \ge dist(f_z, z) - dist(z,v) \ge \frac{2}{3}ecc(z) \ge dist(w,v)$. Thus, when $dist(f_z,v) = dist(w,v)$, f_z is a farthest node of v, contradiction; otherwise, $dist(f_z,v) > dist(w,v)$, w is not the farthest node of v, contradiction. □

Analysis of Approximate Algorithm. Theorem 5.5 shows that after having BFSs performed $|F_1|$ times in IFECC, one can compute the exact eccentricity distribution. Note that $|F_1|$ can be smaller than n but still can be large. The following theorem shows that by performing $|F_2|$, note that $|F_2|$ can be dramatically smaller than $|F_1|$, BFSs in IFECC, one can obtain a reasonably good estimation for the eccentricity of each node.

Theorem 5.6. *Performing BFSs from nodes in F_2 in $O(|F_2|(m+n))$ time computes i) $ecc(v)$ for each node v in F_2 and ii) an estimation $\widetilde{ecc}(v) = \max\{dist_{max}(v,F_2), dist(v,z) + \frac{1}{4}ecc(z)\}$ for each node $v \notin F_2$ such that $\frac{7}{12} \le \frac{\widetilde{ecc}(v)}{ecc(v)} \le \frac{3}{2}$.*

Proof. Consider node $v \notin F_2$.

- If $dist(v,z) \le \frac{1}{3}ecc(z)$, we have $dist(v,F_2) \ge \frac{2}{3}ecc(z)$ while $dist(v,z) + \frac{1}{4}ecc(z) < \frac{2}{3}ecc(z)$. Therefore, $\widetilde{ecc}(v) = dist_{max}(v,F_2) \le ecc(v) \le \max\{\widetilde{ecc}(v), dist(v,z) + \frac{2}{3}ecc(z)\} \le \max\{\widetilde{ecc}(v), ecc(z)\}$. If $\widetilde{ecc}(v) \ge ecc(z)$ then $ecc(v) = \widetilde{ecc}(v)$; otherwise, $\widetilde{ecc}(v) \le ecc(v) \le ecc(z) \le \frac{3}{2}\widetilde{ecc}(v)$.
- If $dist(v,z) \ge \frac{1}{3}ecc(z)$, then: $\frac{7}{12}\max_{u \in V \setminus F_2} dist(v,u)$

$$\le \frac{7}{12}(dist(v,z) + \frac{2}{3}ecc(z)) = \frac{7}{12}dist(v,z) + \frac{5}{36}ecc(z) + \frac{1}{4}ecc(z)$$
$$\le \frac{7}{12}dist(v,z) + \frac{5}{12}dist(v,z) + \frac{1}{4}ecc(z)$$
$$= dist(v,z) + \frac{1}{4}ecc(z) \le ecc(v) + \frac{1}{4}ecc(z)$$
$$\le ecc(v) + \frac{1}{2}ecc(v) = \frac{3}{2}ecc(v). \quad \because ecc(z) \le 2ecc(v)$$

Since $ecc(v) = \max\{\max_{u \in V \setminus F_2} dist(v,u), dist_{max}(v,F_2)\}$ while $\widetilde{ecc}(v) = \max\{dist(v,z) + \frac{1}{4}ecc(z), dist(v,F_2)\}$, we

have: $\frac{7}{12}ecc(v)$

$$
\begin{aligned}
&= \max\{\frac{7}{12}\max_{u\in V\setminus F_2} dist(v,u), \frac{7}{12}dist_{max}(v,F_2)\}\\
&\leq \max\{dist(v,z)+\frac{1}{4}ecc(z), \frac{7}{12}dist_{max}(v,F_2)\}\\
&\leq \max\{dist(v,z)+\frac{1}{4}ecc(z), dist_{max}(v,F_2)\} = \widetilde{ecc}(v)\\
&\leq \max\{\frac{3}{2}ecc(v), dist_{max}(v,F_2)\} = \frac{3}{2}ecc(v).
\end{aligned}
$$

□

The bound given by Theorem 5.6 is considerable given the negative results [28] on approximate solutions; in practice, we have seen a superior performance of kIFECC: on 19 out of 20 real graphs in our experiments, kIFECC can compute the exact eccentricities for all nodes in $|F_2|$ BFS-computations.

Remark. *The above theorems that hold for an arbitrary reference node z show the matching efficiency of our index-free solution in both exact and approximate computations. Note that the size $|F_1|$ and $|F_2|$ is dependent on the reference node z. We will provide a clear view of these parameters on real massive graphs in Section 7 when the highest-degree node is selected as the reference node z.*

6 RELATED WORK

Exact Eccentricity. The straightforward All-Pairs-Shortest-Distance (APSD) algorithms compute the exact eccentricity distribution in quadratic time. APSD-based algorithms cannot scale to large real-world graphs even under optimizations [6, 35]. Negative results show that on an unweighted and undirected sparse graph with n vertices and $m = O(n)$ edges, the diameter (let along the eccentricity distribution) can never be reported in $O(m^{2-\epsilon})$ time, for any constant $\epsilon > 0$, unless the strong exponential time hypnosis can be refuted [28].

In the line of research on practically optimizing exact eccentricity, Henderson [12] applies articulation points and eccentricity bounds; Takes et al. [34] sets priorities on vertices with node degree and pops one vertex a time to update the eccentricity bounds of other nodes until all nodes' eccentricities are determined; PLLECC [19] improves the update rule of [34]. Section 3.2 introduced PLLECC in detail.

Computing the diameter of a graph, which is closely related to the eccentricity computation, can also be hard: Takes el al. [33] proposed a pruning based approach to compute the diameter while Akiba et al. [2] further improve this approach through eccentricity bounds propagation.

Approximate Eccentricity. To alleviate the burden in exact eccentricity computation, approximate eccentricity computation approaches were proposed. We categorize these approaches based on the existence of error bounds.

For the algorithms with error bounds, one may use approximate APSD [1] directly. Roditty et al. [28] presents an algorithm to estimate eccentricity $\widetilde{ecc}(v)$ with time complexity $O(m\sqrt{n\log n})$. The eccentricity $ecc(v)$ is bounded by $[\frac{2}{3}\widetilde{ecc}(v), \frac{3}{2}\widetilde{ecc}(v)]$, for each node v in an undirected and unweighted graph. Chechik et al. [10] further improves the method by transforming the graph to a bounded-degree graph and obtains an algorithm with time complexity $O((m\log m)^{\frac{3}{2}})$ with $ecc(v)$ bounded by $[\widetilde{ecc}(v), \frac{5}{3}\widetilde{ecc}(v)]$.

Table 3: The Data Set Description

Name	Dataset	n	m	r	d	Type
DBLP	DBLP	317,080	1,049,866	12	23	Social
GP	GPlus	201,949	1,133,956	35	70	Social
YOUT	Youtube	1,134,890	2,987,624	12	24	Social
DIGG	Digg	770,799	5,907,132	9	18	social
SKIT	Skitter	1,694,616	11,094,209	16	31	Internet
DBPE	Dbpedia	3,915,921	12,577,253	34	67	Web
HUDO	Hudong	1,962,418	14,419,760	8	16	Web
TPD	UK-Tpd	1,766,010	15,283,718	9	18	Web
FLIC	Flickr	1,624,992	15,476,835	12	24	Social
BAID	Baidu	2,107,689	16,996,139	11	20	Web
TOPC	Topcats	1,791,489	25,444,207	6	11	Web
STAC	Stackoverflow	2,572,345	28,177,464	6	11	Contact
UK02	UK02	18,459,128	261,556,721	23	45	Web
ABRA	Arabic	22,634,275	552,231,867	24	47	Web
IT	IT-2004	41,290,577	1,027,474,895	23	45	Web
TWIT	Twitter	41,652,230	1,202,513,046	13	23	Social
FRIE	Friendster	65,608,366	1,806,067,135	19	37	Social
SK	SK	50,634,118	1,810,050,743	20	40	Web
UK07	UK07	104,288,749	3,293,805,080	56	112	Web
UKUN	UKUN	130,831,972	4,653,174,411	129	257	Web

For approximate algorithms without error bounds, the computation of approximate eccentricities can be more efficient. Takes et al. [34] adjust their exact eccentricity algorithm to terminated early once the diameter and radius can be correctly estimated. Shun [32] adopts a two-stage sampling method to estimate the eccentricities of all nodes. As indicated by their experiments, this sampling algorithm is efficient and provides a high-quality estimation.

Priority Order under BFS-framework. The selection of the source node set S by a priority order plays a vital role under BFS-framework. Henderson [12] iteratively appends S with the node with the largest gap between the upper and lower bounds. Based on [12], various heuristics have been exploited in [33], e.g., appending to S i) the node with the smallest lower bound and the largest upper bound alternatively or ii) the node with the farthest distance to the previously appended node. Akiba, et al. [2] explores the order in appending nodes to S according to in-degree, out-degree, the product of in-degree and out-degree, etc.. The source node set can also be determined based on *random-samples* [28, 32]. Specifically, a set S' of nodes selected from V uniformly at random. S' is used to elect[3] nodes S'' in V that are *likely to be far from other nodes in the graph*. By letting the source node set $S = S' \cup S''$, it is possible to reach $\frac{2}{3}$-approximation if $|S| = \Omega(\sqrt{n}\log n)$.

Other Graph Centrality Measures. There are multiple centrality measures apart from eccentricity centrality. Closeness centrality, the inverse of the sum of shortest distances from a node to all other nodes [26], and betweenness centrality, the fraction of shortest paths between node pairs that pass through the target node [25], indicate the efficiency of a vertex in spreading information to other parts of the graph.[22] summarized many commonly used centrality measures and their applications in various domains.

[3]Different settings were used in [28] and [32] in facilitating this election.

7 EXPERIMENT

This section first introduces the experimental settings and then compares our solution with the competitors. Our statistical analysis on real-world graphs shall echo the theoretical analysis in Section 5.

Algorithms. We compare the following approaches:

- IFECC-16, proposed Algorithm 2 with 16 reference nodes;
- IFECC-1, proposed Algorithm 2 with 1 reference node;
- kIFECC, our proposed approximate Algorithm 3.
- BoundECC [34], the state-of-the-art algorithm for computing exact ED under BFS-framework;
- PLLECC [19], the state-of-the-art algorithm for computing exact ED, parameters were set based on [19];
- kBFS [32], the state-of-the-art algorithm for computing approximate ED. It returns approximate ED given a sample size k.

All algorithms were implemented in C++ and compiled with GNN GCC 4.8.5 with -O3 level optimization. All experiments were conducted on a machine with Intel Xeon 2.4 GHz CPU and 512 GB memory running Linux (Red Hat Linux 4.8.5, 64 bit). We set the cut-off time as 24 hours.

Since both kIFECC and kBFS are under BFS-framework, the quality of the estimation of the eccentricity distribution will be compared under the same number of BFSs that an algorithm can perform. The effectiveness of an approach is evaluated with $\text{Accuracy} = \frac{|\{v|v\in V, \overline{ecc}(v)=ecc(v)\}|}{|V|} \times 100\%$.

Datasets. The experiments were conducted on 20 real-world graphs with various properties, as shown in Table 3. The datasets are from various types of massive networks, including social networks, web graphs, internet topology graphs, and contact networks. The largest graph has more than 4.6 billion edges. All graphs were downloaded from Network Repository[4][30], Stanford Large Network Dataset Collection[5][17], Laboratory for Web Algorithms[6] [7, 8], and the Koblenz Network Collection[7] [16].

7.1 Comparison of Time Consumption

We begin by comparing exact ED computation algorithms regarding time costs. We would like to answer the following questions:

- Q1 What is the advantage of IFECC over other algorithms in terms of computation time?
- Q2 What is the effect of the reference node number r on IFECC?

Exact ED Computation Efficiency. We compare the proposed exact method IFECC to the state-of-the-art exact approach PLLECC. PLLECC contains two main costs: the cost in constructing the index, denoted as PLLECC-PLL, and the cost in computing the eccentricity, denoted as PLLECC-ECC. PLLECC uses 16 reference nodes, as suggested by the settings in [19]. For our method, we denote IFECC with one reference node as IFECC-1 and with 16 reference nodes as IFECC-16. Figure 8 shows the runtime for PLLECC, consists of the time of PLLECC-PLL and PLLECC-ECC, and the time used by our methods IFECC-1 and IFECC-16.

[4]http://networkrepository.com/index.php
[5]http://snap.stanford.edu/data/
[6]http://law.di.unimi.it
[7]http://konect.cc/networks/

Figure 8 shows the superiority of our index-free solution: on the first 12 graphs where PLLECC can complete before the cut-off (24 hours), IFECC-16 is 15 times faster than PLLECC on average. With 1 reference node, IFECC-1 is over 69.8 times faster than PLLECC on the first 12 graphs where PLLECC can compute. On the 12 small graphs, BoundECC completes the computation on the first 11 graphs while it cannot finish the computation on STAC within one day. Moreover, on the graphs where it can finish the computation, BoundECC is 51.6, 558.5, and 2675.3 times slower (on average) than PLLECC, IFECC-16, and IFECC-1, respectively. On the other 8 graphs, IFECC-16 computes the eccentricities for all graphs within 7 hours; IFECC-1 finished all computations within 2 hours. This validates the superior of using only one reference node for IFECC.

Moreover, the index construction time dominates the cost of PLLECC: on the first 12 graphs where PLLECC can complete before the cut-off, the index construction time PLLECC-PLL is more than 41.3× longer than the time PLLECC-ECC to compute the eccentricities. This explains the motivation to eliminate the dependence of the eccentricity calculation on the index.

The Effect of Reference Node Number r. To examine the effect of the reference node number r on the running time of IFECC, r ranges from 1, 2, 4, 8, 16. We denote IFECC under each number r as IFECC-r to report the running time. Figure 9 shows the results.

Figure 9 indicates that the running time of IFECC normally increases with the growth of the reference node number: on all the test graphs, compared with IFECC using one reference node, the running time is more than 1.3, 1.8, 2.8, 4.5 times longer on average when 2, 4, 8, 16 reference nodes are used, respectively. Although on some graphs such as SKIT, the running time of IFECC-2 is a little better than that of IFECC-1, the gap is pretty small: IFECC-1 is less than 1.1 slower than IFECC-2. These results mean that setting reference node number as 1 removes the inconvenient parameter setting process and achieves a superior running time than IFECC with multiple reference nodes at most of the time.

7.2 Comparison of Space Consumption

Then, we compare the space cost of different exact ED calculation algorithms. We would like to answer the following question.

- Q3 What is the advantage of IFECC over the state-of-the-art PLLECC as far as space cost is concerned?

To further verify the superiority of the proposed IFECC, we compare the space cost between PLLECC and IFECC (under a reference node). The space required for IFECC is primarily determined by the graph size, whereas the space required for PLLECC is determined by the graph size plus the index size. We report the memory size consumed by both algorithms at runtime in Figure 10.

On the first twelve graphs where PLLECC can complete the computation, PLLECC requires on average more than 36.6 times more memory space than IFECC, with a maximum of more than 65.4 times (on DBLP). On the eight graphs where PLLECC cannot complete the computation, IFECC consumes less than 40 GB of memory space. Recall that PLLECC requires extra space to load the vast indexes. Conversely, IFECC requires memory space that is linearly related to the graph's size. This further explains why we need to propose new techniques to address PLLECC's dependence on indexes: oversized indexes make PLLECC extremely demanding

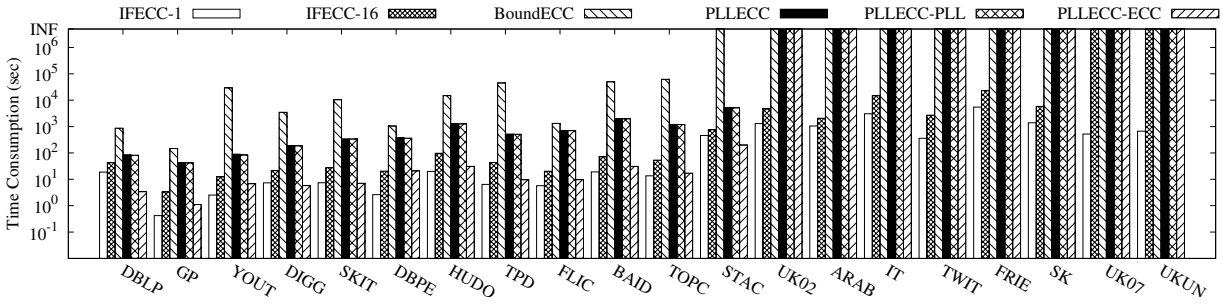

Figure 8: Exact ED Computation Efficiency

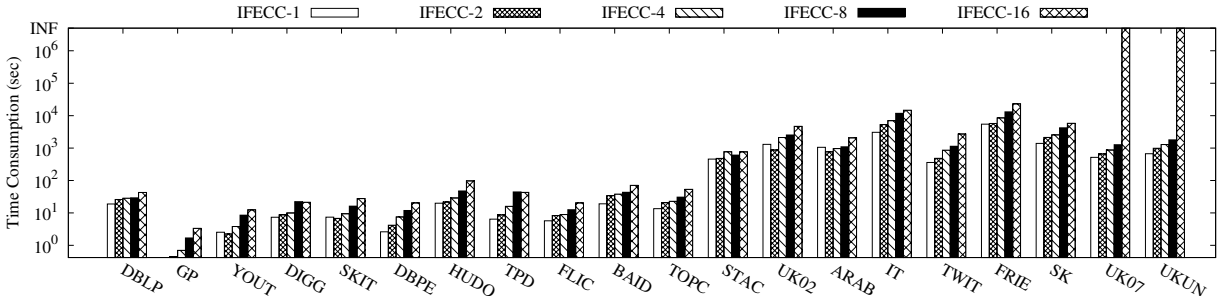

Figure 9: The Effect of Reference Node Number r

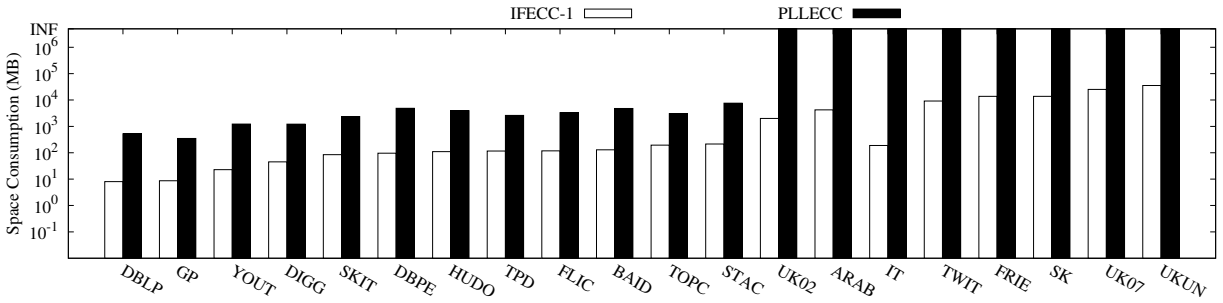

Figure 10: Exact ED Space Comparison

on memory, while our algorithm IFECC does not require much additional memory cost and thus can handle large-scale graphs.

7.3 Approximate ED Computation Efficiency

This experiment compares kIFECC, the approximate version of IFECC, with the state-of-the-art approximate algorithm kBFS. Since both kIFECC and kBFS are under BFS-framework, the cost lies in the number of BFSs performed. Moreover, since both IFECC and kBFS work on the same sample size (i.e., perform BFS), we did not compare the two in terms of time cost, since they have the same time cost. We focus on the accuracy of the two approximation algorithms and try to answer the following question:

Q4 What is the advantage of kIFECC over kBFS in terms of approximation accuracy?

We vary the number k of BFSs from $2^1 = 2$ to $2^7 = 128$. We show the accuracy of each method in Figure 11. On all the tested graphs, the accuracy of kIFECC steadily increases with the increasing of k while that of kBFS does not. For example, when changing k from 2 to 16 on TOPC, kBFS has the accuracy from 27.2% to 8.9%, and then to 99.2%, and finally to 40.2%. This shows an additional instability of kBFS. As the increase of k, kIFECC, adapted from an exact eccentricity computation, will finally report exact eccentricities; in contrast, kBFS, adapted from an approximate approach, may not converge to the exact eccentricities.

7.4 Statistics Analysis

Section 5 analyzed the complexity of IFECC and the effectiveness of kIFECC based on $|F_1|$ and $|F_2|$: $|F_1|$ is enough for exact ED computation and $|F_2|$ is sufficient for approximation ED computation.

Note that the analysis in Section 5 does not depend on a particular choice of the reference node. In practice, however, we find the highest-degree node is a good choice. This is due to the core-periphery structure identified in various networks, especially small-world networks such as social networks and web graphs (see [29] as an entrance). In general, a graph with this property consists of a closely and densely connected core and a sparsely connected periphery. In other words, the periphery is a small set of nodes that are remotely surrounding the core. If one can use the center – the node with the minimum eccentricity – of the graph as the reference node, the corresponding farthest $\frac{1}{3}$ set F_2 would largely overlap the periphery, which leads to a small cardinality $|F_2|$. Since the core is densely connected, the node in the graph with the highest degree

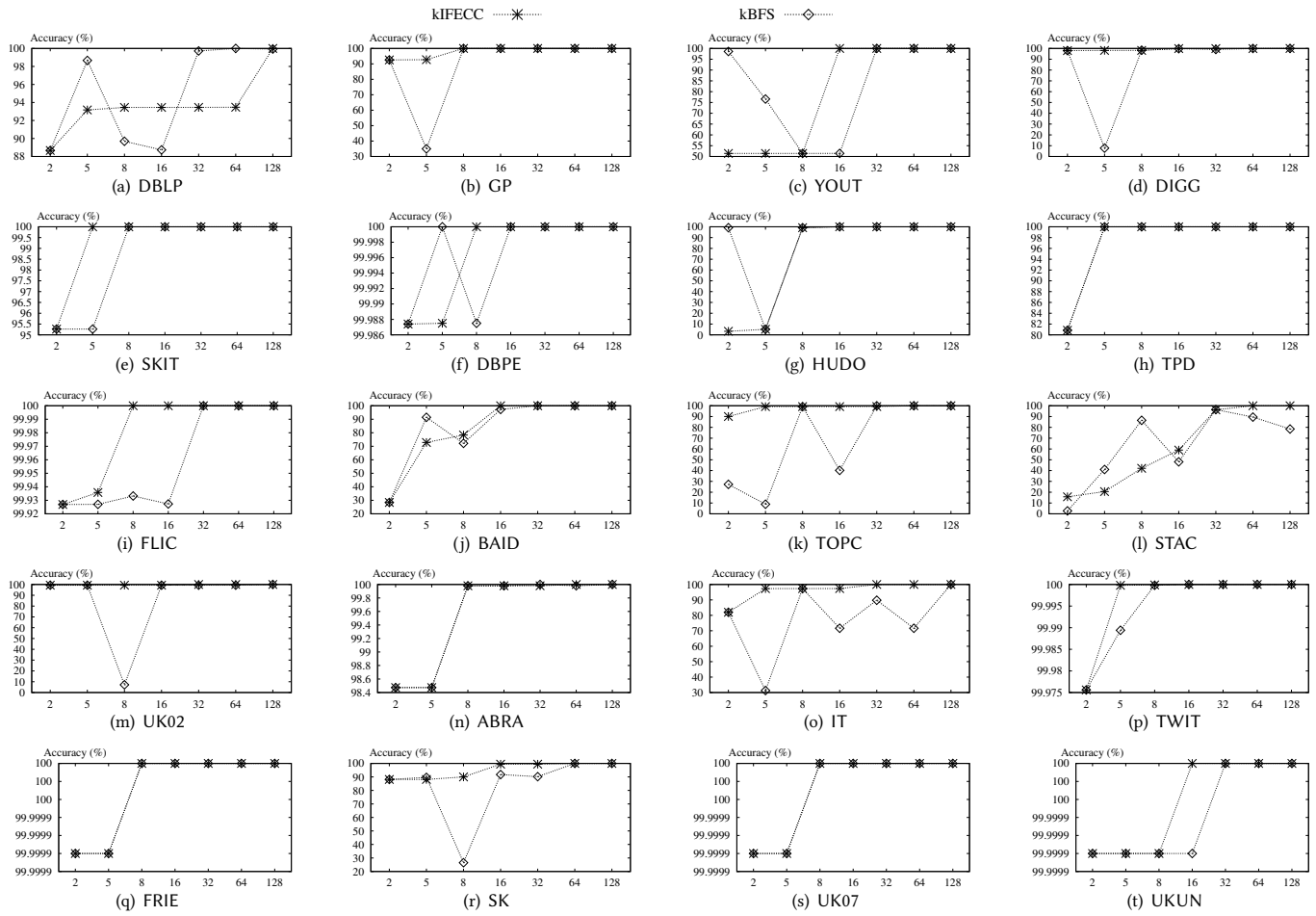

Figure 11: Approximate ED Computation Accuracy

tends to be close to the center of the graph [9]. In other words, by using the highest-degree node as the reference node, one can achieve a small $|F_2|$ and thus efficiency in the bound matching in IFECC. To support this intuition, we ask the following question:

Q5 What is the size of F_1 and F_2 when the highest-degree node is selected for reference on the graphs used in the paper?

In order to answer this question, we show, in Figure 12, $|F_1|$ and $|F_2|$ on 20 real graphs with the highest-degree node the reference node z. Denote by n the number of nodes in the graph. $|F_1|$ is on average $0.1n$ while $|F_2|$ is on average $3.4 \times 10^{-4}n$. The average size of $|F_2|$ is 857.7. In practice, F_2 can do much better approximation than Theorem 5.6; it can compute the eccentricities of an average of 99.999% of nodes in the graph. Besides, on 19 out of 20 data graphs, all nodes' eccentricities are precisely computed.

7.5 Case Study

Stanford Network Analysis Platform (SNAP) [18] is a general-purpose system that provides easy-to-use operations for large graph analysis. One of the important features provided by SNAP is diameter. Diameter dia is a fundamental property of graphs [4]. SNAP estimates the diameter of a graph based on sampling [18]. Specifically, SNAP samples k vertices uniformly at random from V as the sources of BFSs; the maximum eccentricity of the sampled nodes is used as an estimator $\widetilde{dia}$ of the diameter. This study evaluates the effectiveness of SNAP diameter estimation, justifying our effort in replacing the diameter calculation of SNAP with our IFECC.

Exp-1. Effect of Sampling Size on Accuracy. Given $\widetilde{dia}$ and the actual diameter dia, the accuracy of an estimate is defined as $\frac{dia-\widetilde{dia}}{dia} \times 100\%$. To evaluate the performance of SNAP, we varied the sample size used for SNAP from 200, 400, 600, 800, to 1000 (1000 is the number used by SNAP in its code). We calculate the accuracy of the estimated diameter of SNAP for each sample size. Due to space limitations, we only report the results on the four graphs HUDO, TPD, FLIC, and BAID in Fig. 13. Fig. 13 suggests that the average accuracy of SNAP on all graphs is 77.4% which is not satisfactory. Surprisingly, increasing the sample size does not increase the accuracy value: as the sample size varies from 200 to 800, the accuracy on the graph HUDO varies from 75%, 87.5%, 81.3%, and 75%. This implies that the sampling estimator for diameter estimation used by SNAP is unstable.

Exp-2. Comparison with IFECC. To compare IFECC with SNAP, assuming that IFECC requires k BFSs, we control the sample size of SNAP at 20%, 40%, 60%, 80%, and 100% of the number of BFSs used

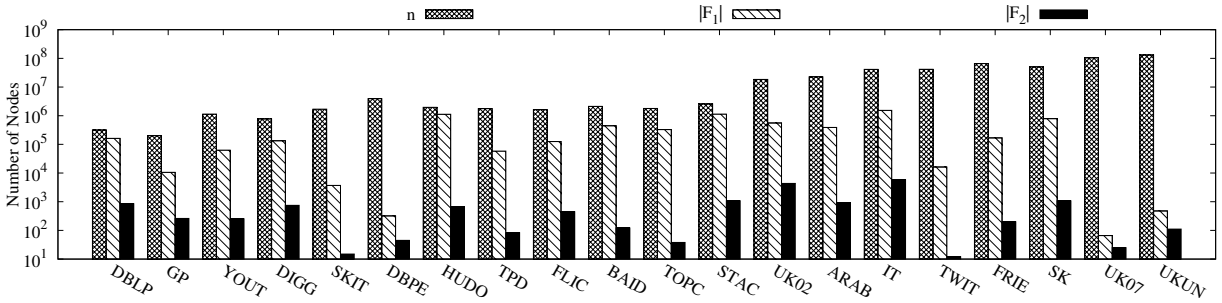

Figure 12: Size of $|F_1|$ and $|F_2|$

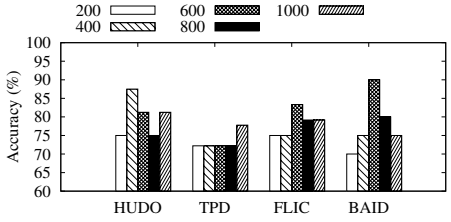

Figure 13: Effect of Sampling Size on Accuracy

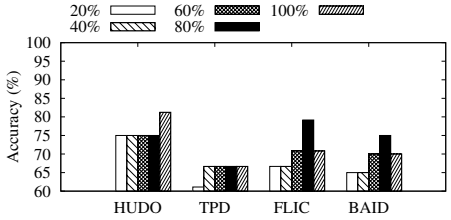

Figure 14: The Comparison with IFECC

by IFECC, and calculate the accuracy of SNAP. We counted the number of BFSs required by our method IFECC to obtain the exact eccentricities: 83 required for HUDO, 26 for TPD, 32 for FLIC, and 61 for BAID. The results are shown in Fig. 14. Fig. 14 suggests that the accuracy of SNAP does not increase with the increase of sample size. Moreover, even when SNAP uses the same sample size as IFECC (i.e., using the same computation time), SNAP has accuracy constantly $\leq 85\%$, while IFECC can obtain the exact diameter. These results strongly suggest that our method can replace the diameter estimation feature of SNAP.

Exp-3. Eccentricity Distribution Plot. This experiment explains why SNAP (random sampling) performs poorly when estimating the diameter. Fig. 15 plots the eccentricity distribution for each graph: the x-axis represents the eccentricity value which is between the radius (minimum eccentricity) and the diameter (maximum eccentricity); the y-axis the number of nodes with that value. It can be observed that the number of nodes whose eccentricity value equals to the diameter is 9 on HUDO, 4 on TPD, 3 on FLIC, and 9 on BAID. The average percentage of these nodes in the whole vertex set is 3.2×10^{-6} which is the probability a random sample can reach the exact diameter. Therefore, it is difficult for SNAP to obtain an accurate estimate of the diameter by sampling. On the contrary, IFECC obtains an accurate diameter (with the exact eccentricity distribution as a by-product) efficiently with an average of 50.5 BFSs on the four graphs. Integrating IFECC into SNAP as a module for diameter and eccentricity distribution computation is a must.

8 CONCLUSIONS

This paper proposes a concise exact eccentricity distribution computation algorithm IFECC. IFECC is up to two orders of magnitude faster than the state-of-the-art algorithm PLLECC and can scale up to billion-scale networks that cannot be handled by PLLECC. Such a performance is, more importantly, theoretically justified. IFECC also derives, as a by-product, a cutting-edge approximate solution for computing the eccentricity distribution.

Acknowledgments. Miao Qiao is supported by Marsden Fund UOA1732, and MBIE Catalyst: Strategic Fund NZ-Singapore Data Science Research Programme UOAX2001. Lu Qin is supported by ARC FT200100787 and ARC DP210101347. Lijun Chang is supported by ARC FT180100256 and ARC DP220103731. Ying Zhang is supported by ARC FT170100128 and ARC DP210101393. Xuemin Lin is supported by ARC DP180103096 and ARC DP170101628.

REFERENCES

[1] Donald Aingworth, Chandra Chekuri, Piotr Indyk, and Rajeev Motwani. 1999. Fast estimation of diameter and shortest paths (without matrix multiplication). *SIAM J. Comput.* 28, 4 (1999), 1167–1181.

[2] Takuya Akiba, Yoichi Iwata, and Yuki Kawata. 2015. An exact algorithm for diameters of large real directed graphs. In *International Symposium on Experimental Algorithms*. Springer, 56–67.

[3] Takuya Akiba, Yoichi Iwata, and Yuichi Yoshida. 2013. Fast exact shortest-path distance queries on large networks by pruned landmark labeling. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*. 349–360.

[4] Réka Albert, Hawoong Jeong, and Albert-László Barabási. 1999. Diameter of the world-wide web. *nature* 401, 6749 (1999), 130–131.

[5] Syed Shafat Ali, Tarique Anwar, and Syed Afzal Murtaza Rizvi. 2020. A revisit to the infection source identification problem under classical graph centrality measures. *Online Social Networks and Media* 17 (2020), 100061.

[6] Paulo Sérgio Almeida, Carlos Baquero, and Alcino Cunha. 2012. Fast distributed computation of distances in networks. In *2012 IEEE 51st IEEE Conference on Decision and Control (CDC)*. IEEE, 5215–5220.

[7] Paolo Boldi, Marco Rosa, Massimo Santini, and Sebastiano Vigna. 2011. Layered label propagation: A multiresolution coordinate-free ordering for compressing social networks. In *Proceedings of the 20th international conference on World wide web*. 587–596.

[8] Paolo Boldi and Sebastiano Vigna. 2004. The webgraph framework I: compression techniques. In *Proceedings of the 13th international conference on World Wide Web*. 595–602.

[9] Arthur Brady and Lenore Cowen. 2006. Compact routing on power law graphs with additive stretch. In *2006 Proceedings of the Eighth Workshop on Algorithm Engineering and Experiments (ALENEX)*. SIAM, 119–128.

[10] 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*. SIAM, 1041–1052.

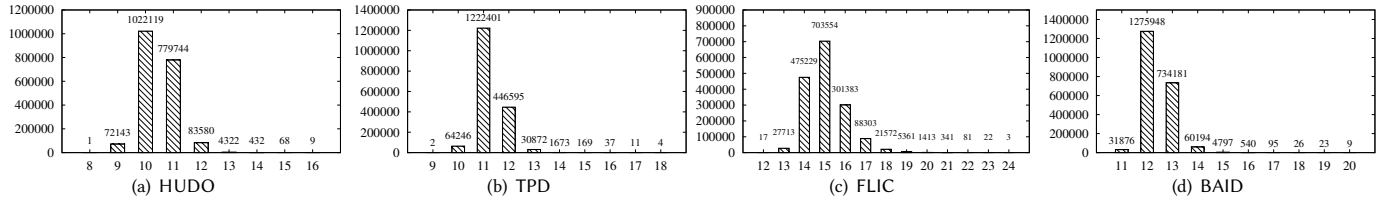

Figure 15: Eccentricity Distribution Plot

[11] Ertugrul N Ciftcioglu, Kevin S Chan, Ananthram Swami, Derya H Cansever, and Prithwish Basu. 2015. Topology control for time-varying contested environments. In *MILCOM 2015-2015 IEEE Military Communications Conference*. IEEE, 1397–1402.

[12] Keith Henderson. 2011. *Opex: Optimized eccentricity computation in graphs*. Technical Report. Lawrence Livermore National Lab.(LLNL), Livermore, CA (United States).

[13] Keita Iwabuchi, Geoffrey Sanders, Keith Henderson, and Roger Pearce. 2018. Computing exact vertex eccentricity on massive-scale distributed graphs. In *2018 IEEE International Conference on Cluster Computing (CLUSTER)*. IEEE, 257–267.

[14] U Kang, Charalampos E Tsourakakis, Ana Paula Appel, Christos Faloutsos, and Jure Leskovec. 2011. Hadi: Mining radii of large graphs. *ACM Transactions on Knowledge Discovery from Data (TKDD)* 5, 2 (2011), 1–24.

[15] Matjaž Krnc, Jean-Sébastien Sereni, Riste Škrekovski, and Zelealem B Yilma. 2020. Eccentricity of networks with structural constraints. *Discussiones Mathematicae Graph Theory* 40, 4 (2020), 1141–1162.

[16] Jérôme Kunegis. 2013. Konect: the koblenz network collection. In *Proceedings of the 22nd International Conference on World Wide Web*. 1343–1350.

[17] Jure Leskovec and Andrej Krevl. 2014. SNAP Datasets: Stanford large network dataset collection.

[18] Jure Leskovec and Rok Sosič. 2016. Snap: A general-purpose network analysis and graph-mining library. *ACM Transactions on Intelligent Systems and Technology (TIST)* 8, 1 (2016), 1–20.

[19] Wentao Li, Miao Qiao, Lu Qin, Ying Zhang, Lijun Chang, and Xuemin Lin. 2018. Exacting Eccentricity for Small-World Networks. In *2018 IEEE 34th International Conference on Data Engineering (ICDE)*. IEEE, 785–796.

[20] Zhaoxing Li, Xianchao Zhang, Hua Shen, Wenxin Liang, and Zengyou He. 2015. A semi-supervised framework for social spammer detection. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 177–188.

[21] YP Liu, B Yan, Jiamou Liu, HY Su, H Zheng, and YJ Cai. 2018. From the Periphery to the Core: Information Brokerage in an Evolving Network. In *IJCAI'18 Twenty-Seventh International Joint Conference on Artificial Intelligence*. International Joint Conferences on Artificial Intelligence.

[22] Linyuan Lü, Duanbing Chen, Xiao-Long Ren, Qian-Ming Zhang, Yi-Cheng Zhang, and Tao Zhou. 2016. Vital nodes identification in complex networks. *Physics Reports* 650 (2016), 1–63.

[23] Damien Magoni and Jean Jacques Pansiot. 2001. Analysis of the autonomous system network topology. *ACM SIGCOMM Computer Communication Review* 31, 3 (2001), 26–37.

[24] Maher Mneimneh and Karem Sakallah. 2003. Computing vertex eccentricity in exponentially large graphs: QBF formulation and solution. In *International Conference on Theory and Applications of Satisfiability Testing*. Springer, 411–425.

[25] Mark EJ Newman. 2005. A measure of betweenness centrality based on random walks. *Social networks* 27, 1 (2005), 39–54.

[26] Kazuya Okamoto, Wei Chen, and Xiang-Yang Li. 2008. Ranking of closeness centrality for large-scale social networks. In *International workshop on frontiers in algorithmics*. Springer, 186–195.

[27] Georgios A Pavlopoulos, Maria Secrier, Charalampos N Moschopoulos, Theodoros G Soldatos, Sophia Kossida, Jan Aerts, Reinhard Schneider, and Pantelis G Bagos. 2011. Using graph theory to analyze biological networks. *BioData mining* 4, 1 (2011), 10.

[28] 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*. 515–524.

[29] Puck Rombach, Mason A Porter, James H Fowler, and Peter J Mucha. 2017. Core-periphery structure in networks (revisited). *SIAM Rev.* 59, 3 (2017), 619–646.

[30] Ryan Rossi and Nesreen Ahmed. 2015. The network data repository with interactive graph analytics and visualization. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*.

[31] M Saravanan and GS Vijay Raajaa. 2012. A graph-based churn prediction model for mobile telecom networks. In *International Conference on Advanced Data Mining and Applications*. Springer, 367–382.

[32] Julian Shun. 2015. An evaluation of parallel eccentricity estimation algorithms on undirected real-world graphs. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 1095–1104.

[33] Frank W Takes and Walter A Kosters. 2011. Determining the diameter of small world networks. In *Proceedings of the 20th ACM international conference on Information and knowledge management*. 1191–1196.

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

[35] Manuel Then, Moritz Kaufmann, Fernando Chirigati, Tuan-Anh Hoang-Vu, Kien Pham, Alfons Kemper, Thomas Neumann, and Huy T Vo. 2014. The more the merrier: Efficient multi-source graph traversal. *Proceedings of the VLDB Endowment* 8, 4 (2014), 449–460.

[36] Turki Turki and Jason TL Wang. 2015. A new approach to link prediction in gene regulatory networks. In *International Conference on Intelligent Data Engineering and Automated Learning*. Springer, 404–415.

[37] Kexiang Xu. 2017. Some bounds on the eccentricity-based topological indices of graphs. *Bounds in Chemical Graph Theory–Mainstreams, Univ. Kragujevac, Kragujevac* (2017), 189–205.

[38] Ke Xiang Xu, Kinkar Ch Das, and Ayse Dilek Maden. 2016. On a novel eccentricity-based invariant of a graph. *Acta Mathematica Sinica, English Series* 32, 12 (2016), 1477–1493.

[39] Guihai Yu and Lihua Feng. 2013. On connective eccentricity index of graphs. *MATCH Commun. Math. Comput. Chem* 69, 3 (2013), 611–628.

APPENDIX

Proof of Lemma 3.1. Given t and its farthest node f_t, v and its farthest node f_v, by the triangular inequality, in Figure 16, from $\triangle(t, v, f_v)$, we have $ecc(v) \leq dist(v,t) + ecc(t)$; from $\triangle(t, v, f_t)$, we have $ecc(t) - dist(v,t) \leq ecc(v)$.

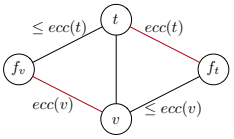

Figure 16: Proof of Lemma 3.1

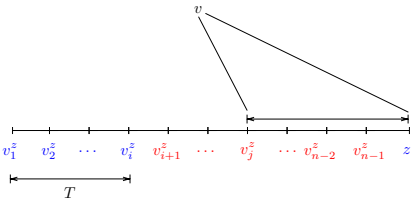

Figure 17: Proof of Lemma 3.3

Proof of Lemma 3.3. By definition, $ecc(v) = \max_{u \in V}(dist(v,u)) = \max_{u \in T \cup \{V \setminus T\}}(dist(v,u)) = \max_{u \in T}(dist(v,u)) + \max_{u \in \{V \setminus T\}}(dist(v,u))$. i) $\underline{ecc}(v) = \max_{u \in T} dist(v,u)$. ii) for $\forall v_j^z \in \{V \setminus T\}$, as given in Figure 17, $dist(v, v_j^z) \leq dist(v,z) + dist(z, v_j^z) \leq dist(v,z) + dist(z, v_i^z)$, and thus $\max_{u \in \{V \setminus T\}}(dist(v,u)) \leq dist(v,z) + dist(z, v_i^z)$. Putting these two parts together derives $ecc(v) \leq \max(\underline{ecc}(v), dist(v,z) + dist(v_i^z, z))$.