Brief Announcement: Eccentricities via Parallel Set Cover

Tal Wagner
MIT
talw@mit.edu

ABSTRACT

The eccentricity of a node in a graph $G(V, E)$ is its maximal shortest-path distance to any other node. Shun (KDD 2015) suggested a simple heuristic for computing all eccentricities in an input graph, based on two-phase parallel BFS from a small sample of nodes. It was shown to outperform state-of-the-art algorithms by up to orders of magnitude. This empirical success stands in apparent contrast to recent theoretical hardness results on approximating all eccentricities (Backurs et al., STOC 2018).

This note aims to formally explain the performance of this heuristic, by drawing a connection to the streaming Set Cover algorithm of Demaine et al. (DISC 2014). We use it to suggest a variant with similar work and depth bounds, which is guaranteed to compute almost all eccentricities exactly, if the graph satisfies a condition we call *small eccentric periphery*. The condition can be ascertained for all real-world graph used in Shun (KDD 2015) and in our experiments. Experimental results demonstrate the validity of the analysis and the empirical advantage of our proposed variant.

1 INTRODUCTION

The eccentricity of a node in a graph is defined as the longest shortest-path distance to any other node reachable from it. Computing all node eccentricities is known to have many useful applications in large-scale graph mining [9, 11, 12]. A simple approach to this task is to perform a breadth-first search (BFS) from each node in a graph. This requires $O(nm)$ work on a graph with n node and m edges, which is prohibitively costly for real-world large-scale graphs. As a result, a large body of research has been dedicated to developing approximate or heuristic algorithms, both in theory [2, 5, 6, 10] and in practice [8, 9, 11].

In a recent work, Shun [11] conducted an empirical study of state-of-the-art algorithms for computing all eccentricities in undirected graphs. The study also included a simple heuristic, referred to henceforth as k-**BFS**$_2$. It samples k uniformly random nodes as *sources*, and computes a full BFS from each source. Then it selects the k nodes with the largest distance to any source, and performs a second phase of BFS with those k nodes as sources. The eccentricity of every node is estimated as its largest distance to any source from either the first or the second phase. The experimental results in [11] showed that k-**BFS**$_2$ performs surprisingly well, outperforming all other methods by large margins. These findings naturally raise questions about a possible formal analysis of this method.

SPAA '19, June 22–24, 2019, Phoenix, AZ, USA

ACM ISBN 978-1-4503-6184-2/19/06.
https://doi.org/10.1145/3323165.3323168

Arguably, the most standard approach to analyzing approximation algorithms is by proving worst-cast multiplicative approximation bounds on the estimates they produce. However, for graph eccentricities, a recent line of work known as *fine-grained complexity* was able to establish complexity-theoretic hardness for improving the current theoretical state-of-the-art algorithms [1, 2]. To circumvent this barrier, we take the path of introducing a structural assumption on the input graph, which can be empirically ascertained for many real-world graphs.

Our Results. We cast the problem of computing all eccentricities as a Set Cover instance with limited access to the input. This draws a close connection between k-**BFS**$_2$ and the streaming Set Cover algorithm of [7]. As a result, we suggest a variant called k-**BFS**$_{\text{SC}}$, with similar work and depth bounds, which is guaranteed to compute the *exact* eccentricities of almost all nodes, as long as the input graph satisfies a property we call *small eccentric periphery*. This property can be ascertained with good parameters for all real-world graphs used in the experiments of [11] as well as in ours. We also give a robust variant which computes $(1-\delta)$-approximate eccentricities for almost all nodes, under a relaxed condition.

k-**BFS**$_{\text{SC}}$ is derived from k-**BFS**$_2$ by a drop-in replacement of its top-k selection step with an off-the-shelf parallel greedy Set Cover algorithm (eg. [3, 4]), leaving the two BFS phases unchanged. The implication is twofold: it serves as evidence that our analysis indeed captures what makes k-**BFS**$_2$ work in practice, and also that our proposed variant is plausible for implementation and practical use. The latter point is particularly relevant since k-**BFS**$_2$ is highly successful in practice, and our experiments show that our variant can significantly improve its performance.

Preliminaries and Notation. We assume that the input graph $G(V, E)$ has n labeled nodes and is undirected, unweighted and connected. The shortest-path distance between two nodes $v, u \in V$ is denoted by $\Delta(v, u)$. The eccentricity of v is defined as $e(v) = \max_{u \in V} \Delta(v, u)$.

2 k-BFS$_2$ BY SET COVERING

Our variant of k-**BFS**$_2$ is called k-**BFS**$_{\text{SC}}$ and is given in Algorithm 1. It provably returns accurate eccentricity estimates for graphs that satisfy a property we call *small eccentric periphery*.

Definition 2.1. A graph $G(V, E)$ has *eccentric periphery of size* κ if κ is the smallest integer such that there exists $U \subset V$ of size κ, such that for every $v \in V$, $e(v) = \Delta(v, u)$ for some $u \in U$.

Put simply, this property states that all node eccentricities can be realized as distances to a subset of κ nodes. As a warm-up, one may observe that a path, star, clique and perfect binary tree all have eccentric peripheries of size 2, regardless of their size, whereas a cycle on n nodes has eccentric periphery of size n if n is even, or $\frac{1}{2}(n+1)$ if n is odd.

Obviously if the input graph has eccentric periphery of size κ, then there is a realization of k-**BFS**$_2$ with any $k \geq \kappa$ that computes

Algorithm 1 : k-**BFS**$_{\text{SC}}$

Input: Graph $G(V, E)$, integer $k > 0$
Output: Eccentricity estimate $\hat{e}(v)$ for every $v \in V$

$U_1 \leftarrow k$ uniformly random nodes *// Phase 1*
foreach $u \in U_1$:
  Compute a BFS started at u
foreach $v \in V$:
  $\tilde{A}_v \leftarrow \{u \in U_1 : e(u) = \Delta(u, v)\}$ *// based on BFS results*
$\mathcal{I} \leftarrow$ Set Cover instance with elements U_1 and sets $\{\tilde{A}_v\}_{v \in V}$
$\mathcal{C} \leftarrow$ cover for $\mathcal{I}$ *// using parallel greedy Set Cover algorithm*
$U_2 \leftarrow$ the set of nodes such that $\mathcal{C} = \{A_u : u \in U_2\}$ *// Phase 2*
foreach $u \in U_2$:
  Compute a BFS started at u
foreach $v \in V$:
  return $\hat{e}(v) = \max_{u \in U_1 \cup U_2} \Delta(v, u)$ *// based on BFS results*

all eccentricies exactly, given the right choice of BFS sources. The question is how to identify those sources, and more specifically, how does k-**BFS**$_2$ apparently succeed in finding them.

To answer this, we point out that if the all-eccentricities problem is viewed as a Set Cover instance, then k-**BFS**$_2$ is seen to be closely related to the streaming Set Cover algorithm of [7]. By making this connection exact, we obtain k-**BFS**$_{\text{SC}}$ with the following guarantee.

Theorem 2.2. *Suppose $G(V, E)$ has eccentric periphery of size at most κ. Let $\varepsilon > 0$. Then for $k = \tilde{O}(\varepsilon^{-1}\kappa \log n)$, k-**BFS**$_{\text{SC}}$ runs in $O(k \cdot |E|)$ expected work and $O(\text{diam}(G) \cdot \log n + \log^3(kn))$ expected depth, and with high probability computes the exact eccentricities of all but an ε-fraction of the nodes in V.*

The applicability of the above result hinges on whether the small eccentric periphery property occurs in real-world graph. Here it is worth noting that the property can be ascertained for all 8 real-world graphs considered in [11],[1] as a byproduct of the experiments therein. In particular, whenever k-**BFS**$_2$ computes a $(1-\varepsilon)$-fraction of the eccentricities exactly, it certifies that the eccentric periphery has size at most $\varepsilon n + 2k$. Thus, 7 of the graphs in [11], containing between $1M$ to $4M$ nodes, were shown to have eccentric periphery size of only few thousands (between 0.1% to 0.6% of their nodes). For two of them the size is as small as 128 nodes. The 8th graph has eccentric periphery containing 4.4% of its nodes. These are upper bounds obtained as a byproduct, and it remains possible that the true parameters are even smaller.

2.1 Set Cover Formulation

Recall that in the Set Cover problem, we are given a set of elements $\mathcal{E}$ and a collection of subsets $\mathcal{S} \subset 2^{\mathcal{E}}$. We call $\mathcal{C} \subset \mathcal{S}$ a *cover* if $\mathcal{E} \subset \cup_{A \in \mathcal{C}} A$. The goal is to find a cover of minimum size.

Computing all eccentricites can be cast as a Set Cover instance as follows. For every $u \in V$ define $A_u = \{v \in V : e(v) = \Delta(v, u)\}$, i.e., A_u is the subset of nodes whose eccentricity is attained by their distance to u. The Set Cover instance is formed by the elements $\mathcal{E} = V$ and sets $\mathcal{S} = \{A_u : u \in V\}$. Given $U \subset V$, define $e_U(v) = \max_{u \in U} \Delta(u, v)$ for every $v \in V$, and consider $e_U(v)$ as an estimate for $e(v)$. Consider the set of sets $\mathcal{C}_U = \{A_u : u \in U\}$. We see that if $\mathcal{C}_U$ covers v (i.e., $v \in A_u$ for some $A_u \in \mathcal{C}_U$), then $e(v) = e_U(v)$. Therefore, computing all eccentricities exactly as $\{e_U(v) : v \in V\}$ reduces to solving the above Set Cover instance with a cover $\mathcal{C}_U$. Furthermore, the optimal cover size is precisely the eccentric periphery size, as per Definition 2.1.

Let us highlight the non-standard computational constraints of this Set Cover setting, that arise if it is to be used for computing all eccentricities. Given the index $u \in V$ of a set A_u, it is prohibitive to compute which elements are contained in A_u, since that already requires computing all eccentrities. Given an element $v \in V$, it is expensive but non-prohibitive to compute which subsets contain it, since that requires a single full BFS started at v. Hence we can afford it for only a small number of elements.

2.2 Relation to DIMV

While we are not aware of any Set Cover algorithms that were explicitly designed for these constraints, there is in fact one that meets them: the streaming Set Cover algorithm of [7], referred to henceforth as DIMV. This is somewhat incidental, and indeed other streaming Set Cover algorithms do not meet these constraints. Another interesting fact is that k-**BFS**$_2$ turns out to be closely related to DIMV, as we explain next.

DIMV is a combination of two modules: The *set sampling* module simply includes random sets in the output cover. The *element sampling* module chooses a small random sample of elements, and computes a cover only for the sample using an offline black-box algorithm (eg. greedy). Note that set sampling is a vanilla module that need not know anything about which sets cover which elements, while the more informed element sampling module only needs to know which sets cover the elements in the sample. Thus both of them meet the model constraints specified above. The key observation is that k-**BFS**$_2$ corresponds to a combination of set and element sampling, as follows:

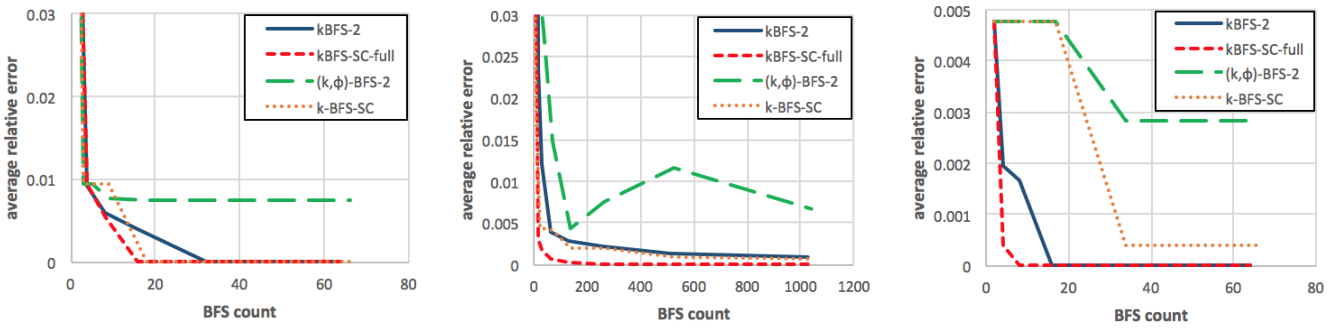
1. The first phase runs a BFS from each node u in a random sample U_1. This implicitly computes which subsets $\{A_v\}_{v \in V}$ cover u.
2. The second phase computes U_2 as the k nodes that maximize $e_{U_1}(v)$. This is akin to computing a cover $\mathcal{C}_{U_2}$ of U_1.[2]

Thus we interpret the top-k selection step in k-**BFS**$_2$ as a heuristic Set Cover step. Indeed, the node v^* with the largest estimate $e_{U_1}(v^*)$ satisfies $e(u^*) = \Delta(u^*, v^*)$ for some $u^* \in U$, and thus A_{v^*} covers u^*. However, the node in V with the second-largest $e_{U_1}(v)$ estimate might redundantly cover u^* again, and so on, ultimately leaving some elements uncovered.

To avoid such degeneracy, and make the connection to DIMV exact (which would allow us to leverage its formal analysis), all we need is to replace the covering heuristic by an actual Set Cover algorithm. Fortunately, this is a well-studied problem in parallel computing. In particular, we can use the parallel greedy algorithm of [3], which guarantees an approximation factor of $O(\log |\mathcal{E}|)$

[1] This count does not include four additional graphs for which the true eccentricities were not computed in [11] due to their large size.

[2] Note that the final estimates of k-**BFS**$_2$ are $e_{U_1 \cup U_2}(v)$, which correspond to the cover $\mathcal{C}_{U_1} \cup \mathcal{C}_{U_2}$ rather than just $\mathcal{C}_{U_2}$. Hence each $u \in U_1$ plays a dual role of a sample element to cover in the second phase, and a set A_u in the final cover. The first phase is thus seen to concurrently function as both element sampling and set sampling, though our analysis will only rely on the element sampling role.



(a) Oregon-1_{010526} **(b)** p2p-Gnutella30 **(c)** loc-Brightkite

Graph Name	Properties				Eccentric periphery size	k	k-**BFS**$_2$		k-**BFS**$_{\text{SC–full}}$		Cover size (ϕ)	(k, ϕ)-**BFS**$_2$		k-**BFS**$_{\text{SC}}$	
	Nodes	*Edges*	*Diam.*	*Avg.* $e(v)$			*CR*	*ARE*	*CR*	*ARE*		*CR*	*ARE*	*CR*	*ARE*
Oregon-1_{010526}	11,174	23,409	10	7.15	≤ 32	16	0.969	0.004	1	0	2	0.945	0.007	0.999	$9 \cdot 10^{-5}$
						64	1	0	1	0	2	0.945	0.007	0.999	$9 \cdot 10^{-5}$
p2p-Gnutella30	36,646	88,303	11	8.69	≤ 1024	16	0.802	0.022	0.988	0.001	3	0.731	0.032	0.961	0.004
						64	0.960	0.004	0.998	$2 \cdot 10^{-4}$	5	0.869	0.015	0.964	0.004
loc-Brightkite	56,739	212,945	18	11.75	≤ 16	16	1	0	1	0	2	0.958	0.004	0.958	0.004
						64	1	0	1	0	4	0.968	0.003	0.999	$4 \cdot 10^{-4}$

(essentially optimal unless P = NP) and has been tested for implementation [4]. This leads to k-**BFS**$_{\text{SC}}$ and to Theorem 2.2.

One may ask whether such degeneracy in the covering step of k-**BFS**$_2$ in fact shows up in practice, or in other words, whether we expect k-**BFS**$_{\text{SC}}$ to improve over k-**BFS**$_2$ empirically. Indeed, this exact phenomenon has been recently discussed in [8], who report observing it in many large real-world graphs, and take heuristic measures to mitigate its adverse effect on the accuracy of k-**BFS**$_2$. Our Set Cover based approach avoids it in a principled manner.

3 EXPERIMENTS

As input graphs, we use three real-world graphs from the Stanford Network Analysis Project (available at http://snap.stanford.edu/data/). In each graph we use only the largest connected component (which contains almost all nodes), and treat all edges as undirected.

For an informed comparison, we introduce two more algorithmic variants. Note that while k-**BFS**$_2$ uses k sources in the second phase, k-**BFS**$_{\text{SC}}$ uses only ϕ sources, where ϕ is the cover size computed by the greedy Set Cover algorithm based on the first phase. It is often the case that ϕ is much smaller than k. To equate the total work, we introduce the variant k-**BFS**$_{\text{SC–full}}$, which is similar to k-**BFS**$_{\text{SC}}$ except that in the second phase it uses the greedy strategy to choose a possibly redundant set cover of size exactly k. Thus, both k-**BFS**$_2$ and k-**BFS**$_{\text{SC–full}}$ use a total of $2k$ BFS sources, k in each phase, differing in how the second phase sources are chosen. k-**BFS**$_{\text{SC}}$ uses a total of $k+\phi$ sources, k in the first phase and ϕ in the second. To complete the picture, we also include the variant (k, ϕ)-**BFS**$_2$, which uses k sources in the first phase and ϕ sources in the second, chosen by the same rule as k-**BFS**$_2$, i.e., by a top-ϕ selection step. For this variant, ϕ is an external parameter which we set according to the results of k-**BFS**$_{\text{SC}}$, for the sake of comparison between them.

The x-axis in the attached figures counts the number of BFS invocations, as a proxy for the total work. The y-axis measures their average relative error. All algorithms were run with $k = 2^i$ for $i = 0, 1, \ldots, 10$, though for visual clarity, the x-axes are truncated when all plots have stabilized. Some additional numbers are given in the attached table. In addition to the average relative error (ARE), it contains the correctness ratio (CR), which is the fraction of nodes whose eccentricity was computed exactly, and an upper bound on the eccentric periphery size of each graph.

The results show that k-**BFS**$_{\text{SC–full}}$ dominates the other algorithms and converges faster to near-zero error. The greedy Set Cover selection rule for the BFS sources in the second phase is significantly preferrable to top-k selection, improving the accuracy by up to an order of magnitude.

REFERENCES

[1] A. Abboud, V. V. Williams, and J. Wang. Approximation and fixed parameter subquadratic algorithms for radius and diameter in sparse graphs. In *SODA*, 2016.
[2] A. Backurs, L. Roditty, G. Segal, V. V. Williams, and N. Wein. Towards tight approximation bounds for graph diameter and eccentricities. In *STOC*, 2018.
[3] G. E. Blelloch, R. Peng, and K. Tangwongsan. Linear-work greedy parallel approximate set cover and variants. In *SPAA*, 2011.
[4] G. E. Blelloch, H. V. Simhadri, and K. Tangwongsan. Parallel and i/o efficient set covering algorithms. In *SPAA*, 2012.
[5] M. Cairo, R. Grossi, and R. Rizzi. New bounds for approximating extremal distances in undirected graphs. In *SODA*, 2016.
[6] S. Chechik, D. Larkin, L. Roditty, G. Schoenebeck, R. Tarjan, and V. V. Williams. Better approximation algorithms for the graph diameter. In *SODA*, 2014.
[7] E. D. Demaine, P. Indyk, S. Mahabadi, and A. Vakilian. On streaming and communication complexity of the set cover problem. In *DISC*, 2014.
[8] K. Iwabuchi, G. Sanders, K. Henderson, and R. Pearce. Computing exact vertex eccentricity on massive-scale distributed graphs. In *IEEE International Conference on Cluster Computing (CLUSTER)*, 2018.
[9] U. Kang, C. E. Tsourakakis, A. P. Appel, C. Faloutsos, and J. Leskovec. Hadi: Mining radii of large graphs. *ACM TKDD*, 5(2):8, 2011.
[10] L. Roditty and V. Vassilevska Williams. Fast approximation algorithms for the diameter and radius of sparse graphs. In *STOC*, 2013.
[11] J. Shun. An evaluation of parallel eccentricity estimation algorithms on undirected real-world graphs. In *KDD*, 2015.
[12] F. W. Takes and W. A. Kosters. Computing the eccentricity distribution of large graphs. *Algorithms*, 6(1):100–118, 2013.