2018 IEEE 59th Annual Symposium on Foundations of Computer Science

Sublinear Algorithms for Local Graph Centrality Estimation

Marco Bressan
Sapienza Università di Roma
Roma, Italy
bressan@di.uniroma1.it

Enoch Peserico
Università degli Studi di Padova
Padova, Italy
enoch@dei.unipd.it

Luca Pretto
Università degli Studi di Padova
Padova, Italy
pretto@dei.unipd.it

***Abstract*—We study the complexity of local graph centrality estimation, with the goal of approximating the centrality score of a given target node while exploring only a sublinear number of nodes/arcs of the graph and performing a sublinear number of elementary operations. We develop a technique, that we apply to the PageRank and Heat Kernel centralities, for building a low-variance score estimator through a local exploration of the graph. We obtain an algorithm that, given any node in any graph of m arcs, with probability $(1-\delta)$ computes a multiplicative $(1\pm\epsilon)$-approximation of its score by examining only $\tilde{O}\big(\min\big(m^{2/3}\Delta^{1/3}d^{-2/3},\, m^{4/5}d^{-3/5}\big)\big)$ nodes/arcs, where Δ and d are respectively the maximum and average outdegree of the graph (omitting for readability poly(ϵ^{-1}) and polylog(δ^{-1}) factors). A similar bound holds for computational cost. We also prove a lower bound of $\Omega\big(\min\big(m^{1/2}\Delta^{1/2}d^{-1/2},\, m^{2/3}d^{-1/3}\big)\big)$ for both query complexity and computational complexity. Moreover, our technique yields a $\tilde{O}(n^{2/3})$-queries algorithm for an n-node graph in the access model of Brautbar et al. [1], widely used in social network mining; we show this algorithm is optimal up to a sublogarithmic factor. These are the first algorithms yielding worst-case sublinear bounds for general directed graphs and any choice of the target node.**

***Keywords*-graph centrality; sublinear algorithms; local algorithms; query and computational complexity; random walks; PageRank; heat kernel**

I. Introduction

Computing graph centralities efficiently is essential to modern network analysis. With the advent of web and social networks, a typical scenario involves massive graphs of millions or even billions of nodes and arcs. On these input graphs, traditional approaches such as Monte Carlo simulations and algebraic techniques are often impractical – if not entirely useless – since their cost can scale linearly or superlinearly with the size of the graph. On the other hand, in many cases all one needs is an approximation of the score of a few nodes of interest. Ideally, this humbler goal can be met by *local graph algorithms* that explore only a small fraction of the graph – using significantly fewer computational resources. This approach has been used effectively, for example, in the context of graph clustering algorithms [2], [3], [4].

In this paper we address the problem of locally approximating the centrality score of a node in a graph, focusing on the PageRank and Heat Kernel centralities. PageRank [5] is a classic graph centrality measure with a vast number of applications including local graph clustering [2], trendsetter identification [6], spam filtering [7], link prediction [8] and many more (see [9] and [10]); it has been named one of the top 10 algorithms in data mining [11]. Heat Kernel [12] can be seen as a variant of PageRank that satisfies the heat equation, with applications from biological network analysis [13], [14] to the solution of local linear systems [15]; like PageRank, Heat Kernel has a long and successful history in local graph clustering algorithms [12], [16], [17], [18], [19]. The inputs to our problem are a directed graph G, a target node $v \in G$, and approximation parameters $\epsilon, \delta \in (0,1)$. The output is a value $p(v)$ that, with probability $(1-\delta)$, is a multiplicative $(1\pm\epsilon)$-approximation of the centrality score $P(v)$ of v. The goal is to compute $p(v)$ by accessing only a sublinear portion of G's nodes and arcs, and using a sublinear number of elementary operations. In other words, we look for algorithms with sublinear query cost and sublinear computational cost.

In the case of PageRank, the local approximation of $P(v)$ has a history dating back over a decade [20], [21], [22], [23], [24], [25], [26], [27], [28], [29], [30]. It is well understood that PageRank can be seen as a fast-mixing random walk, which allows one to approximate all scores larger than p with $\tilde{\Theta}(1/p)$ operations [26], [27], [31], [32]. However, all but $o(n)$ nodes in an n-node graph have score $O(1/n)$, which means the cost is $\Omega(n)$ for essentially every target node in G. A complementary approach is to estimate $P(v)$ by exploring the graph backwards from v towards its ancestors [20], [22], [23], [24]; however, this approach in isolation is subject again to a $\Omega(n)$ lower bound [23], [24], [25], [33]. One can also combine the two techniques, which basically amplifies the information given by the random walks [28], [29], [30]. However, so far this has yielded sublinear upper bounds only in expectation over v [28], [30] or for target nodes of low degree in undirected graphs [29]; no sublinear bounds are known for general directed graphs and arbitrary choices of the target node.

A similar scenario holds for Heat Kernel, where research is focused on heat kernel diffusions and their connection to Cheeger's constants and local graph clustering [18], [19],

2575-8454/18/$31.00 ©2018 IEEE
DOI 10.1109/FOCS.2018.00073

[34], [35], but from which no useful bound can be derived for the problem of approximating $P(v)$.

Our Results. In this paper we present algorithms for approximating $P(v)$ with fully sublinear worst-case query cost and computational cost. Our algorithms work for general directed graphs and any choice of the target node v, for both PageRank and Heat Kernel; they can in principle be used for other random-walk-based centralities too, with complexity bounds depending on the choice of parameters. For computational complexity, we use the standard RAM model. For query complexity, we primarily use the standard graph access model of [36], [37], where the number of queries is essentially the number of arcs examined. More precisely, let m be the number of arcs of G, and Δ and d be respectively its maximum and average *out*degree. We prove[1]:

Theorem 1. *The query complexity of computing with probability* $(1-\delta)$ *a* $(1\pm\epsilon)$*-approximation of* $P(v)$ *is* $\tilde{O}\big(\min\big(m^{2/3}\Delta^{1/3}d^{-2/3},\, m^{4/5}d^{-3/5}\big)\big)$.

Theorem 2. *The computational complexity of computing with probability* $(1-\delta)$ *a* $(1\pm\epsilon)$*-approximation of* $P(v)$ *is* $\tilde{O}\big(\min\big(m^{3/4}\Delta^{1/4}d^{-3/4},\, m^{6/7}d^{-5/7}\big)\big)$.

The two bounds derive from essentially the same algorithm by optimizing query cost and computational cost *separately*. One can however keep both costs simultaneously sublinear – for example according to the bound of Theorem 2, since computational complexity is an upper bound to query complexity. In general, one can trade between the two. Our results show one can *always* break through the $\Theta(m)$ complexity barrier by polynomial factors, while all previous algorithms do no better than $O(m)$ unless one looks at special cases (e.g. disconnected graphs or nodes with large score). For example, in graphs with $\Delta = O(\text{poly}\log(n))$, which is a reasonable assumption for many social networks, our algorithms access only $\tilde{O}(n^{2/3})$ nodes and arcs, or perform only $\tilde{O}(n^{3/4})$ operations. Approximating $P(v)$ via random walks, instead, requires $\Theta(n)$ queries and operations, as does the technique of [28] if for example v has indegree $\Theta(n)$. In fact our algorithms are sublinear in n, too, unless $m = \Theta(n^2)$.

Our second contribution are lower bounds on the query and computational complexity of approximating $P(v)$, for both PageRank and Heat Kernel. Formally, we prove:

Theorem 3. $\Omega\big(\min\big(m^{1/2}\Delta^{1/2}d^{-1/2},\, m^{2/3}d^{-1/3}\big)\big)$ *queries and elementary operations are in general required to approximate* $P(v)$ *within a factor* $O(1)$ *with probability* $\Omega(1)$.

[1]Unless necessary, for readability we hide in the $O()$ notation multiplicative factors depending only on the approximation parameters ϵ, δ, which in our case are (mildly) polynomial in ϵ^{-1} and polylogarithmic in δ^{-1}. Similarly, the $\tilde{O}()$ notation hides factors polylogarithmic in n, that in most of our bounds are actually sublogarithmic.

Although these lower bounds do not match the upper bounds, at the very least they show one cannot solve the problem with e.g. only poly $\log(n)$ queries and/or operations. We also note that, unless $\Delta = \tilde{\Theta}(d)$ or $m = \tilde{\Theta}(n^2)$, our lower bounds are asymptotically larger than the $\tilde{O}(m^{1/2})$ upper bounds given by [28] for a uniform choice of v (see Section II), proving such bounds cannot hold for *every* choice of v. We leave open the question of whether one can tighten our upper bounds, our lower bounds, or both.

Our final contribution are almost-tight query complexity bounds for approximating $P(v)$ under the model of [1], widely used in the field of large graph mining [24], [26], [38], [39], [40], [41]. The model provides a (powerful) query that returns, in one shot, all the incoming and outgoing arcs of the queried node; therefore, n queries are always sufficient. Equivalently, one can think of the query as revealing one of the n rows and one of the n columns of the adjacency matrix of G. We obtain the first algorithm requiring only $o(n)$ queries, and one that we prove optimal up to sublogarithmic factors:

Theorem 4. *In the model of [1], the query complexity of computing with probability* $(1-\delta)$ *a* $(1\pm\epsilon)$*-approximation of* $P(v)$ *is* $\tilde{O}(n^{2/3})$. *Moreover,* $\Omega(n^{2/3})$ *queries and elementary operations are in general required to approximate* $P(v)$ *within a factor* $O(1)$ *with probability* $\Omega(1)$.

Organization of the paper. The rest of this introduction deals with notation and definitions. Section II summarizes the state of the art. Section III provides a detailed walk-through of the ideas and techniques behind our results. Section IV sketches the proof of the lower bounds. Section V and Section VI provide a short overview of, respectively, the adaptation for Heat Kernel and the porting to the model of [1]. All details omitted, including pseudocode and proofs, can be found in the full version of this paper [42].

A. Preliminaries and notation

We denote by $G = (V, A)$ the directed input graph, and by $n = |V|$ and $m = |A|$ the number of its nodes and arcs. For simplicity we assume n is known; however one can estimate it by sampling $O(\sqrt{n})$ random nodes from G (see [43]), which leaves our bounds unchanged. If $(u, w) \in A$ we say u is a parent of w and w is a child of u, and we write $u \to w$. We denote by $in(u)$ and $out(u)$ the indegree and outdegree of u, and by $d = \frac{m}{n}$ and $\Delta = \max_{u \in G} out(u)$ the average and maximum outdegree of G. We denote by $G[u, u', \ldots]$ the subgraph of G induced by the set of nodes $\{u, u', \ldots\}$. For simplicity we assume G has no dangling nodes (u is dangling if $out(u) = 0$); this assumption makes the discussion much lighter and can be easily lifted [42].

We denote by $\mathbf{A}$ the normalized (row-stochastic) adjacency matrix of G, so $a_{ij} = \frac{1}{out(i)}$ if $(i, j) \in A$ and $a_{ij} = 0$

otherwise. The PageRank score vector $\mathbf{p}$ is then defined as:

$$\mathbf{p} = (1-\alpha)\sum_{\tau \geq 0} \alpha^\tau \mathbf{f}\,\mathbf{A}^\tau \tag{1}$$

where the *damping factor* α is a constant strictly less than 1, and $\mathbf{f}$ is a stochastic *preference vector*. The Heat Kernel score vector is the analogue of PageRank under exponential damping:

$$\mathbf{p} = e^{-\alpha}\sum_{\tau \geq 0} \frac{\alpha^\tau}{\tau!}\mathbf{f}\,\mathbf{A}^\tau \tag{2}$$

In both cases $\|\mathbf{p}\|_1 = 1$, i.e. the scores form a probability distribution. $P(v)$ is the entry of $\mathbf{p}$ associated to node v. In our case, we set $\mathbf{f}$ to the uniform distribution $[\frac{1}{n} \dots \frac{1}{n}]$. Note that this implies $P(v) = \Omega(\frac{1}{n})$ for all v. For PageRank, a useful equality that can be derived from (1) is $P(v) = \frac{1-\alpha}{n} + \sum_{u \to v} P(u)\frac{\alpha}{out(u)}$ (a slightly more involved relationship holds for Heat Kernel). Our full proofs make use of an equivalent formulation of $P(v)$ in terms of random walks, too (see [42]).

For computational complexity, we adopt the standard RAM model. For query complexity, we adopt the standard model of [36], [37], where G is accessed through an oracle answering the following *queries*: INDEG(u), that returns $in(u)$; OUTDEG(u), that returns $out(u)$; PARENT(u, i), that returns the i-th parent of u or NIL if $in(u) < i$; CHILD(u, i), that returns the i-th child of u or NIL if $out(u) < i$. We also allow the use of a query JUMP() [1], that returns a node chosen uniformly at random from G. Note that every call to one of these queries counts as an elementary operation, so query complexity is a lower bound to computational complexity. As mentioned before, we obtain bounds in the model of [1], too; the allowed queries are NEIGH(u), that returns the parents and the children of u, and JUMP().

II. Related work

Most existing work concerns the local approximation of PageRank. The problem itself was introduced in [20], and in its many forms has attracted considerable attention since then [21], [22], [23], [24], [25], [26], [27], [28], [29], [30], [31], [44].

A first set of papers [26], [27], [31], [44] addressed the problem of sketching the score vector of G efficiently through random walks using JUMP() and CHILD(). Since one needs $\Omega(1/P(v))$ samples in order to hit v, this approach is subject to a $\Omega(1/P(v))$ query and computational complexity lower bound [26], [27], which means $\Omega(n)$ for essentially every node in G. A second set of papers [22], [45] focused on estimating how much each node $u \in G$ contributes to $P(v)$ (in terms of random walks, how easily one reaches v from u). These algorithms do not use JUMP(), and explore G backwards from v towards its ancestors. Although such algorithms can in principle be used to estimate $P(v)$, the lack of JUMP() makes them subject to a query complexity lower bound of $\Omega(n)$ (for Monte Carlo) or $n - o(n)$ (for Las Vegas) [24], [25], [46].

Combining random walks and backward exploration, a set of recent papers [28], [29], [30], [47] proved novel results on the local approximation of Personalized PageRank (PPR) [5] and related problems. The key idea is to take into account the random walks that hit the ancestors of v, reducing the necessary number of samples. Their main result is FAST-PPR, an algorithm that approximates the PPR score of a given node within a factor $(1 \pm \epsilon)$, whenever the score is larger than a given $\delta > 0$. FAST-PPR has running time $\tilde{O}(\sqrt{d/\delta})$ in expectation over a uniform random choice of $v \in G$, which for plain PageRank means $\delta = \Theta(1/n)$ and an expected running time of $\tilde{O}(m^{1/2})$. The framework of FAST-PPR was subsequently used to compute Markov chain multi-step transition probabilities [47], with similar average-case guarantees, and for local PageRank approximation on undirected graphs [29], with a worst-case running time $O(n^{1/2}\, in(v)^{1/2})$. These results are encouraging, but at the same time suggest that obtaining *worst-case* sublinear upper bounds for the general case is nontrivial. As in [28], [29], [30], [47], in this paper we combine random walks with backward exploration. However, we do not use the backward exploration of [22], [45], but instead we introduce two novel tools, *subgraph estimators* and *weighted subgraph estimators*, that make it easier to control both the variance of our estimator and the cost of its construction. Thanks to these tools, we give the first worst-case sublinear upper bounds that hold for any directed graph G and all nodes $v \in G$. Incidentally, we show that the average-case $\tilde{O}(m^{1/2})$ upper bound of [28] cannot hold in the worst case.

As for Heat Kernel, existing local approximation algorithms focus on the so-called *diffusion* – essentially, the distribution of the random walk from a given seed node – due to its relationship with local low-conductance cuts and local graph clustering [18], [19], [34], [35]. There exists work on efficiently computing the action of the matrix exponential on vectors [35], [48], [49], but no useful bounds can be derived for the local approximation of $P(v)$.

Finally, we would mention recent work on the local approximation of the stationary probability of a target state v in a Markov Chain [47], [50], [51], and on the local approximation of a single entry of the solution vector of a linear system [52], [53]. The local approximation of $P(v)$ is a specific but nontrivial case of both, and we hope that our techniques may serve as an entry point for future developments in those directions.

III. Sketch of the proofs

This section gives a detailed step-by-step sketch of the algorithms and proofs behind Theorem 1 and Theorem 2. We focus on PageRank; the case of Heat Kernel is analogous, but requires lengthy technical adaptations that are briefly presented in Section V. Porting our results to the model of [1] (Theorem 4) requires adapting our algorithms as well, as described in Section VI. Before proceeding, let us outline the main ideas and techniques in the order they appear in the proof sketch.

A. Random walk sampling. As a basic primitive we sample the nodes $u \in G$ with probability equal to their score $P(u)$, by emulating the random walk for $O(1)$ expected steps per sample. We can then associate to each $u \in G$ an indicator random variable χ_u with expectation $P(u)$. Estimating $P(v)$ by repeated sampling of χ_v is possible, but requires $\Omega(n)$ samples in the worst case.

B. Subgraph estimators. Given any induced subgraph H of G containing v, by expressing $P(v)$ recursively in terms of the scores of its ancestors we define a *subgraph estimator* $p_H(v)$ satisfying $\mathbb{E}[p_H(v)] = P(v)$. Formally, $p_H(v)$ is a weighted sum of the χ_u of the nodes u bordering H (akin to the "blanket sets" of [28]). We can take a sample of $p_H(v)$ using the random walk. Unfortunately, the coefficients of the χ_u can be unbalanced and one of them can dominate, making $p_H(v)$ behave essentially as a single χ_u, and one may still need $\Omega(n)$ samples to estimate $P(v)$.

C. Weighted estimators. To bypass these limitations we introduce *weighted estimators*, which are weighted averages of sequences of subgraph estimators $p_{G_0}(v), p_{G_1}(v), \ldots$. We build a weighted estimator $p_k(v)$ on the sequence of subgraphs $H = G_0, G_1, \ldots, G_k$ that we visit by exploring G starting from v. This requires moving from G_{i-1} to G_i by choosing a new node u_i and then *expanding* it by examining its parents and their outdegrees. The advantage over subgraph estimators is that $p_k(v)$ exploits the χ_u associated to the nodes *inside* G_k too, while adding a degree of freedom in the choice of the estimator's coefficients.

D. Building a perfect estimator. We prove how, by carefully choosing which nodes $u_0, \ldots, u_k$ to expand, and the weights of the subgraph estimators $p_{G_0}(v), \ldots, p_{G_k}(v)$, we can build a "perfect" weighted estimator $p_k(v)$ that behaves essentially as the plain sum of k non-positively correlated indicator random variables χ_u. This means the variance of $p_k(v)$ is drastically lower than that of every single $p_{G_i}(v)$. By standard concentration bounds we can then show that $\ell = \Theta(\frac{n}{k})$ samples of $p_k(v)$ suffice to estimate $P(v)$ within our approximation guarantees.

E. Blacklisting heavy nodes. We then adapt the construction of $p_k(v)$ so as to avoid expanding nodes with high indegree. First, we take ℓ random-walk samples; this reveals all "heavy" nodes having score $\tilde{\Omega}(\ell^{-1})$, together with good estimates of their score. The intuition is that high-indegree nodes are heavy. Second, if while building $p_k(v)$ we encounter a heavy node, instead of expanding it we plug its estimate directly into $p_k(v)$. We then show that the resulting estimator $q_k(v)$ preserves the approximation guarantees.

F. Indegree inequalities. We then bound the total number of queries t used to build $q_k(v)$ by bounding the indegree $in(u_i)$ of each node we expand. To this end we give inequalities that bound $in(u_i)$ from above in terms of m, d, Δ and of $P(u_i)$. It follows that $in(u_i)$ must be small since we only expanded nodes with small $P(u_i)$. The resulting bound is used to minimize the total queries over all the phases (blacklisting, building, sampling).

G. Approximate estimators. We then turn to computational cost. This is dominated by the construction of $q_k(v)$ and, more precisely, by the computation of the subgraph estimators $p_{G_i}(v)$. We show that we just need an additive $\frac{\epsilon}{n}$-approximation of the coefficients of $p_{G_i}(v)$, which one can compute using $\Theta(\ln(n/\epsilon))$ sparse matrix-vector products on the normalized adjacency matrix of G_i. The resulting bound is again used to minimize the total computational cost over all the phases (blacklisting, building, sampling).

A. Random walk sampling

Our first ingredient is a primitive to sample v with probability $P(v)$. We employ SAMPLENODE() [42], a simple routine originally introduced in [31], [44], which emulates PageRank's random walk using JUMP() and STEP() queries. SAMPLENODE() returns node u with probability $P(u)$; moreover, it has expected query and computational cost $O(1)$, thus ℓ invocations have query and computational cost $O(\ell)$ with high probability. For each $u \in G$ we then let χ_u be the indicator random variable of the event that SAMPLENODE() returns u, so that $\mathbb{E}[\chi_u] = P(u)$ and we can sample χ_u with $O(1)$ expected operations. Clearly, the χ_u are non-positively correlated since they indicate mutually exclusive events. The random variables χ_u will appear throughout all the rest of the paper. Note that one could naively estimate $P(v)$ by repeatedly invoking SAMPLENODE(), but this requires $\Omega(n)$ queries since in the worst case $P(v) = O(\frac{1}{n})$.

B. Subgraph estimators

Our second ingredient is the *subgraph estimator* of $P(v)$. Recall from Section I-A that $P(v)$ can be written recursively as $P(v) = \frac{1-\alpha}{n} + \sum_{u \to v} P(u) \frac{\alpha}{out(u)}$. If we now replace $P(u)$ with χ_u, since $\mathbb{E}[\chi_u] = P(u)$ we obtain a random variable $p(v)$ with expectation exactly $P(v)$. More in general, instead of v and its parents, we can consider an induced subgraph $H \subseteq G$ containing v, and the nodes of $G \setminus H$ having outgoing arcs that end in H. Formally, we have:

Definition 1. *The* frontier *of an induced subgraph H of G is the set of arcs $F(H) = \{(u, w) \in A : u \notin H, w \in H\}$. We say a node u is* on *the frontier of H if there is an arc $(u, w) \in F(H)$ – in which case we abuse notation slightly and write $u \in F(H)$.*

The intuition is that $F(H)$ collects the part of $P(v)$ due to the random walks originating in $G \setminus H$. Indeed, any random walk starting in $G \setminus H$ must go through some $u \in F(H)$ before hitting v. This intuition can be formalized in the following lemma (see [42] for a proof):

Lemma 5. *For any induced subgraph H of G and $v \in H$:*

$$P(v) = c_H + \sum_{u \in F(H)} P(u) \cdot c_H(u) \tag{3}$$

where c_H and $c_H(u)$ depend only on H, on $F(H)$, and on the outdegrees of the nodes that are in H or on $F(H)$.

By replacing once again $P(u)$ with χ_u, we obtain:

Definition 2. *The* subgraph estimator *of $P(v)$ given by H is the random variable:*

$$p_H(v) = c_H + \sum_{u \in F(H)} \chi_u \cdot c_H(u) \tag{4}$$

By construction of H we always have $c_H > 0$, $c_H(u) > 0$. Now, since the χ_u are non-positively correlated, we can use standard concentration bounds on $p_H(v)$. The strength of the bounds depends on the coefficients $c_H(u)$: if one of them dominates, then $p_H(v)$ behaves essentially like a single χ_u and we may still need $\Omega(n)$ samples. This happens for example if H contains only v and if all parents of v have high outdegree, save one parent having outdegree 1. In fact there may be no H making the coefficients balanced, and furthermore we know the coefficients only after knowing H. Perhaps surprisingly, however, we can overcome all these limitations by simply combining subgraph estimators into a weighted sum.

C. Weighted estimators

The notion of *weighted estimator* of $P(v)$ is a cornerstone of our technique. Informally, a weighted estimator is just the weighted average of a set of subgraph estimators. Formally:

Definition 3. *Let $G_0, \dots, G_k$ be a set of induced subgraphs of G, each one containing v. Let $\beta_0, \dots, \beta_k$ be nonnegative reals such that $\sum_{i=0}^{k} \beta_i = 1$. The* weighted estimator *given by $G_0, \dots, G_k$ with weights $\beta_0, \dots, \beta_k$ is the random variable:*

$$\begin{aligned} p_k(v) &= \sum_{i=0}^{k} \beta_i \, p_{G_i}(v) \\ &= \sum_{i=0}^{k} \beta_i \Big(c_{G_i} + \sum_{u \in F(G_i)} \chi_u \cdot c_{G_i}(u) \Big) \end{aligned} \tag{5}$$

Let us see how to employ weighted estimators. Start by setting $u_0 = v$ and $G_0 = G[u_0]$, and expand u_0 so as to learn $F(G_0)$ and the outdegrees of all $u \in F(G_0)$. We can then compute the coefficients of $p_{G_0}(v)$. Choose some $u_1 \in F(G_0)$, let $G_1 = G[u_0, u_1]$, and expand u_1 so as to learn $F(G_1)$ and the outdegrees of all $u \in F(G_1)$. We can then compute the coefficients of $p_{G_1}(v)$. Now choose some $u_2 \in F(G_1)$, and so on. We obtain a sequence of progressively larger subgraphs $G_0, \dots, G_k$ giving subgraph estimators $p_{G_0}(v), \dots, p_{G_k}(v)$. By choosing weights $\beta_0, \dots, \beta_k$ we finally obtain our weighted estimator $p_k(v)$.

Let us take a closer look at $p_k(v)$. For each u appearing in the expression of $p_k(v)$ we let $j(u) = \min\{j : u \in F(G_j)\}$. By rearranging the expression of (5) so as to collect together the terms pertaining to the same node, we obtain:

$$\begin{aligned} p_k(v) = \sum_{i=0}^{k} \beta_i \, c_{G_i} \; + \sum_{i=1}^{k} \chi_{u_i} \cdot \sum_{j=j(u_i)}^{i-1} \beta_j \, c_{G_j}(u_i) \\ + \sum_{u \in F(G_k)} \chi_u \cdot \sum_{j=j(u)}^{k} \beta_j \, c_{G_j}(u) \end{aligned} \tag{6}$$

The first summation of (6) gathers constant terms (containing no random variables). The second summation involves the nodes $u_1, \dots, u_k$ that we have expanded and are thus in G_k. The third summation involves the nodes that are still on $F(G_k)$. Our focus is on the second summation. Our next step is to show that, by carefully choosing $u_1, \dots, u_k$ and $\beta_0, \dots, \beta_k$, we can make the coefficients of $\chi_{u_1}, \dots, \chi_{u_k}$ *all identical*, leading to (strong) concentration bounds. This holds even though the coefficients of every single subgraph estimator $p_{G_i}(v)$ may be heavily unbalanced.

D. Building a perfect weighted estimator

Let us rewrite (6) more compactly. We write the first summation, $\sum_{i=0}^{k} \beta_i \, c_{G_i}$, as c_k. In the second summation, we write the coefficient $\sum_{j=j(u_i)}^{i-1} \beta_j c_{G_j}(u_i)$ as $c_k(u_i)$. In the third summation, we write the coefficient $\sum_{j=j(u)}^{k} \beta_j c_{G_j}(u)$ as $c_k(u)$. Our weighted estimator is then:

$$p_k(v) = c_k + \sum_{i=1}^{k} \chi_{u_i} c_k(u_i) + \sum_{u \in F(G_k)} \chi_u \, c_k(u) \tag{7}$$

We call the two summations of (7) respectively *inner summation* and *frontier summation*. Similarly, we call their coefficients *inner coefficients* and *frontier coefficients*. Ideally, we would like to make all the coefficients identical. We settle for a slightly less ambitious target:

Definition 4. *We say that the weighted estimator $p_k(v)$ is* perfect *if:*

$$\begin{aligned} &c_k(u_i) = c_k(u_{i'}) \quad && \forall i, i' \in \{1, \dots, k\} \\ &c_k(u_i) \geq c_k(u) \quad && \forall i \in \{1, \dots, k\}, \; \forall u \in F(G_k) \end{aligned}$$

The heart of our algorithm consists in building a perfect weighted estimator $p_k(v)$ by exploring as little of G as possible. In fact, we show one can always build such a $p_k(v)$ by expanding exactly $k+1$ nodes.

Let us sketch the construction. Suppose by inductive hypothesis that we have built a perfect weighted estimator $p_{k-1}(v)$ on subgraphs $G_0, \dots, G_{k-1}$ using weights $\beta_0, \dots, \beta_{k-1}$. Therefore, by Definition 4 the inner coefficients $c_{k-1}(u_i)$ are all identical. Recall the expression of $p_{k-1}(v)$ given by (7) together with the definitions of $c_{k-1}(u_i)$ and $c_{k-1}(u)$. Now note that every frontier coefficient $c_{k-1}(u)$ contains β_{k-1}, while the inner coefficients $c_{k-1}(u_i)$ contain only $\beta_0, \dots, \beta_{k-2}$. Hence, if we change β_{k-1} while rescaling $\beta_0, \dots, \beta_{k-2}$ so that the overall sum remains 1, we can change the ratio between inner coefficients and frontier coefficients. In particular, we can choose β_{k-1} so that the largest (breaking ties arbitrarily) of the frontier coefficients matches the value of the inner coefficients. The node u associated to this largest frontier coefficient is the next node u_k to expand. Indeed, we can build $p_k(v)$ from $p_{k-1}(v)$ by moving the term relative to u_k from the frontier summation to the inner summation, expanding u_k, and then adding to the frontier summation the new terms relative to the parents of u_k that are on $F(G_k)$. We then simply set the new weight $\beta_k = 0$. One can check that the weighted estimator $p_k(v)$ is again perfect. Formally, one can prove [42]:

Lemma 6. *One can build a perfect weighted estimator $p_k(v)$ using $\sum_{i=0}^{k}(1 + 2\,in(u_i))$ queries.*

Before bounding the query cost and computational cost of building $p_k(v)$, we analyze its concentration around $P(v)$.

E. Concentration bounds for $p_k(v)$

We bound the probability that $|p_k(v) - P(v)| > \epsilon P(v)$, to obtain the desired guarantees. Recall again the expression of $p_k(v)$ given by (7). By Definition 4, if $c = c_k(u_i)$ is the value of the inner coefficients, then the random variable $c^{-1}(p_k(v) - c_k)$ is a sum of non-positively correlated binary random variables with coefficients in $[0, 1]$. Furthermore, the probability that $p_k(v)$ deviates by more than a multiplicative $(1 \pm \epsilon)$ from $P(v)$ is bounded by the probability that $c^{-1}(p_k(v) - c_k)$ deviates by more than a multiplicative $(1 \pm \epsilon)$ from $\mathbb{E}[c^{-1}(p_k(v) - c_k)]$. By standard concentration bounds we then obtain [42]:

$$\Pr\Big[\frac{|p_k(v) - P(v)|}{P(v)} > \epsilon\Big] < 2e^{-\epsilon^2 \mathbb{E}[c^{-1}(p_k(v) - c_k)]/3} \tag{8}$$

Crucially, we bound from below $\mathbb{E}[c^{-1}(p_k(v) - c_k)]$ by restricting $p_k(v)$ to the inner summation:

$$\mathbb{E}[c^{-1}(p_k(v) - c_k)] \geq \mathbb{E}\Big[c^{-1}\sum_{i=1}^{k}\chi_{u_i} c_k(u_i)\Big] = \mathbb{E}\Big[\sum_{i=1}^{k}\chi_{u_i}\Big] \tag{9}$$

Finally, since $\mathbb{E}[\chi_{u_i}] = P(u_i) \geq \frac{1-\alpha}{n}$ for any u_i, we obtain $\mathbb{E}[c^{-1}(p_k(v) - c_k)] \geq \frac{1-\alpha}{n}k$. Now, if we take ℓ independent samples of $p_k(v)$, the expectation obviously grows to $\frac{1-\alpha}{n}k\ell$. Therefore, if $p_k^{\ell}(v)$ is the average of $p_k(v)$ over ℓ samples, by (8) we obtain:

$$\Pr\Big[\frac{|p_k^{\ell}(v) - P(v)|}{P(v)} > \epsilon\Big] < 2e^{-\epsilon^2(1-\alpha)k\ell/3n} \tag{10}$$

To obtain a multiplicative $(1 \pm \epsilon)$-approximation of $P(v)$ with probability $(1 - \delta)$ we then simply choose $k\ell = \Theta(n\epsilon^{-2}\ln(1/\delta))$, or equivalently $\ell = \Theta(\frac{n}{k}\epsilon^{-2}\ln(1/\delta))$.

The core of our algorithm is complete: we know how to build a perfect weighted estimator $p_k(v)$ and how many samples ℓ to take. We now turn to bounding the query and computational cost of building $p_k(v)$.

F. Blacklisting heavy nodes

We must bound the query cost of building $p_k(v)$, which by Lemma 6 equals $\sum_{i=0}^{k}(1 + 2\,in(u_i))$. Unfortunately, this might be $\Theta(m)$, e.g. if $m = O(n)$ and some u_i has $in(u_i) = \Theta(n)$. We thus modify our algorithm so as to avoid expanding nodes with high indegree – in fact, more generally, with high score.

First of all, we take ℓ samples via SAMPLENODE(). For each $u \in G$ let $s(u)$ be the fraction of times u is returned, and let $B = \{u \in G : s(u) \geq \frac{16\ln(2n/\delta)}{\epsilon^2\ell}\}$. We call *blacklisted* the nodes in B. By standard arguments one can show [42]:

Lemma 7. *With probability at least $1 - \frac{\delta}{2}$:*
1. $\{u \in G : P(u) \geq \frac{25\ln(2n/\delta)}{\epsilon^2\ell}\} \subseteq B$
2. $(1 - \epsilon)P(u) \leq s(u) \leq (1 + \epsilon)P(u)$ *for all* $u \in B$

If $v \in B$, by Lemma 7 $s(v)$ with probability $1 - \frac{\delta}{2}$ is a $(1 \pm \epsilon)$-approximation of $P(v)$. We can then just return $s(v)$ and stop. If instead $v \notin B$, we build an estimator $q_k(v)$ similar to $p_k(v)$ but that avoids expanding nodes of B. The idea is the following. Suppose by inductive hypothesis that for some $k \geq 1$ we have built an estimator $q_{k-1}(v)$ which has the same form as $p_k(v)$ but is such that $u_i \notin B$ for all $i = 0, \dots, k-1$. We then build $q_k(v)$ as in the original construction, but choosing u_k among the nodes on $F(G_{k-1})$ that are not in B (if there are no such nodes then we simply stop). Therefore we set β_{k-1} so that, among all nodes on $F(G_{k-1})$ not in B, the largest coefficients equals the inner coefficients. As a consequence, in $q_k(v)$ we have $u_i \notin B$ for all $i = 0, \dots, k$; while every $u \in B$ encountered during the construction is left indefinitely on the frontiers $F(G_{j(u)}), \dots, F(G_k)$. Formally, $q_k(v)$ has the form:

$$q_k(v) = c_k + \sum_{i=1}^{k}\chi_{u_i}c_k(u_i) + \sum_{\substack{u\in F(G_k)\\ u\notin B}}\chi_u c_k(u) + \sum_{\substack{u\in F(G_k)\\ u\in B}}\chi_u c_k(u) \tag{11}$$

Now, by construction, if we consider only the first three terms of (11) then $q_k(v)$ is a perfect weighted estimator (Definition 4). We can thus apply the same bounds of $p_k(v)$ (Subsection III-E). For the last summation, instead, we just replace each χ_u with $s(u)$, obtaining by Lemma 7 a multiplicative $(1 \pm \epsilon)$-approximation of the expectation of the whole sum with probability $1 - \frac{\delta}{2}$. Later, at sampling time, we can simply discard any sampled $u \in B$. By a union bound, then, we get for $q_k(v)$ the same guarantees of $p_k(v)$. We now conclude our query cost bounds by bounding $in(u_i)$ for $i = 0, \dots, k$.

G. Indegree inequalities

Recall that building $q_k(v)$ requires $\sum_{i=0}^{k}(1 + 2\,in(u_i))$ queries (by Lemma 6), and that we can assume $P(u_i) < \frac{25\ln(2n/\delta)}{\epsilon^2 \ell}$ for all $i = 0, \dots, k$ (by Lemma 7). We now bound $in(u_i)$ in terms of $P(u_i)$ and the parameters of the graph. The intuition is that $P(u_i)$ is directly proportional to $in(u_i)$, and inversely proportional to the outdegrees of u_i's parents, which in turn are tied to m, d, Δ. Formally, we show [42]:

Lemma 8. *For any $u \in G$:*

$$in(u) = O\Big(\min\Big(\frac{m\Delta P(u)}{d}, \frac{mP(u)^{1/2}}{d^{1/2}}\Big)\Big) \qquad (12)$$

Let $t = \sum_{i=0}^{k}(1 + 2\,in(u_i))$. Now bound each term $in(u_i)$ as per Lemma 8, then in turn bound $P(u_i)$ as per Lemma 7. This leads to:

$$t = O\Big(\min\Big(\frac{km\Delta\ln(n/\delta)}{\epsilon^2 d\ell}, \frac{km}{\epsilon}\Big(\frac{\ln(n/\delta)}{d\ell}\Big)^{1/2}\Big)\Big) \qquad (13)$$

Now set $k = \frac{n\ln(1/\delta)}{\epsilon^2 \ell} = \frac{m\ln(1/\delta)}{\epsilon^2 d\,\ell}$ as required by the concentration bounds of Section III-E. Then, since the blacklisting and sampling phases require $O(\ell)$ queries, impose $\ell = t$ to minimize the total asymptotic number of queries $\ell + \ell + t$ of our algorithm. We obtain:

$$\begin{aligned}\ell = O\big(\min\big(&m^{2/3}\Delta^{1/3}d^{-2/3}\ln(n/\delta)^{1/3}\ln(1/\delta)^{1/3}\epsilon^{-4/3},\\ &m^{4/5}d^{-3/5}\ln(n/\delta)^{1/5}\ln(1/\delta)^{2/5}\epsilon^{-6/5}\big)\big)\end{aligned} \qquad (14)$$

which becomes $\tilde{O}\big(\min\big(m^{2/3}\Delta^{1/3}d^{-2/3}, m^{4/5}d^{-3/5}\big)\big)$ if we disregard factors depending only on ϵ and δ. For ℓ sufficiently large, we can invoke a union bound on the probability that building $q_k(v)$ requires more than ℓ queries (through Lemma 7) and on the probability of $q_k^{\ell}(v)$ deviating excessively from $P(v)$ (10), proving Theorem 1.

H. Approximate estimators

We finally bound the computational cost of our algorithm. The blacklisting and sampling phase take $O(\ell)$ operations. The computationally intensive part is the construction of $q_k(v)$, or equivalently $p_k(v)$, which is dominated by the computation of the coefficients. We can however show one just needs a good approximation of these coefficients. Formally, we have:

Definition 5. *$p'_k(v)$ is an* additive $\bar{\epsilon}$-approximation *of $p_k(v)$ if $\big|\mathbb{E}[p'_k(v)] - \mathbb{E}[p_k(v)]\big| \le \bar{\epsilon}$.*

By choosing $\bar{\epsilon} = \Theta(\frac{\epsilon}{n})$ arbitrarily small we can then bring $\mathbb{E}[p'_k(v)]$ and $\mathbb{E}[p_k(v)]$ within a multiplicative factor arbitrarily close to 1. We then build $p'_k(v)$ as we did for $p_k(v)$, so as to obtain a perfect weighted estimator and keep all the concentration guarantees (with concentration around $\mathbb{E}[p'_k(v)]$ instead of $P(v)$). In exchange for the additive approximation we can build $p'_k(v)$ faster than $p_k(v)$. To this end, when computing the coefficients of $p_{G_i}(u)$, we ignore the contribution given by paths of length $\omega(\ln(n/\epsilon))$; we only compute the contribution of paths of length $O(\ln(n/\epsilon))$, with $O(\ln(n/\epsilon))$ matrix-vector multiplications on the adjacency matrix of G_k. Since this matrix has at most t nonzero entries, the computation takes $O(t\ln(n/\epsilon))$ operations. Formally, we prove [42]:

Lemma 9. *One can build an additive $\bar{\epsilon}$-approximation $p'_k(v)$ of $p_k(v)$ in $O(kt\ln(1/\bar{\epsilon}))$ operations.*

With our choice $\bar{\epsilon} = \Theta(\frac{\epsilon}{n})$ this implies we can build $p'_k(v)$ in $O(kt\ln(n/\epsilon))$ operations. We then use the bound $t = \tilde{O}\big(\min\big(\frac{km\Delta}{d\,\ell}, km(\frac{1}{d\,\ell})^{1/2}\big)\big)$ given by (13), and we set $\ell = \frac{m\ln(1/\delta)}{\epsilon^2 d\,k}$ for the concentration bounds. We obtain $t = \tilde{O}\big(\min\big(\Delta k^2, k^{3/2}m^{1/2}\big)\big)$. It follows that $q'_k(v)$ can be built using $\tilde{O}\big(\min\big(\Delta k^3, k^{5/2}m^{1/2}\big)\big)$ operations. Finally, we minimize the overall computational cost by setting this quantity equal to the $O(\ell) = O(\frac{m}{d\,k})$ bound of the blacklisting and sampling phase. We obtain:

$$\ell = \tilde{O}\big(\min\big(m^{3/4}\Delta^{1/4}d^{-3/4}, m^{6/7}d^{-5/7}\big)\big) \qquad (15)$$

which proves Theorem 2.

Remarks.

Obliviousness to m, d, Δ. Our algorithms, as defined above, need explicit knowledge of m, d, Δ; this requirement is easily waived. For the number of queries we proceed as follows. Choose an initial value for ℓ and perform all phases on a budget of ℓ queries. This is straightforward for blacklisting and sampling; for building $q_k(v)$, simply stop as soon as expanding u_i would outstrip the budget. If at sampling time $c^{-1}(q_k^{\ell}(v) - c_k) = \Theta(\epsilon^{-2}\ln(1/\delta))$, by standard concentration bounds $c^{-1}(q_k^{\ell}(v) - c_k)$ and thus $q_k^{\ell}(v)$ are within a factor $(1 \pm \epsilon)$ of their expectation with probability $(1 - \delta)$ (see Section III-E). If not, we double ℓ and repeat. A similar argument holds for the computational cost.

Portability of our techniques. Our techniques can be used in other settings satisfying some simple requirements. First, we need a primitive to sample nodes with probability proportional to their score. Second, we need $P(v)$ to be a (positive) linear combination of the scores of v's ancestors. We can then build and sample a perfect estimator $p_k(v)$. Third, we need a lower bound on each $P(u)$, or at least on $\sum_{i=1}^{k} P(u_i)$, so that we can attain concentration for $p_k(v)$. Fourth, $P(u)$ must be increasing with $in(u)$, so that we can blacklist nodes with large indegree. These ingredients are present at least partially when dealing with centralities such as Katz's [54], or with Markov chains. What bounds can be obtained in these and other cases is left for future research.

IV. Lower bounds

We sketch the proof of Theorem 3. For every function $d(n) \in \Omega(1) \cap O(n)$ we construct a family of n-node graphs with average degree $d \in \Theta(d(n))$; each of them sports a node v such that any algorithm needs $\Omega\big(\min\big(m^{1/2}\Delta^{1/2}d^{-1/2},\, m^{2/3}d^{-1/3}\big)\big)$ queries to output an $O(1)$-approximation of $P(v)$ with non-negligible probability. The structure of the generic graph G is shown in Fig. 1. The target node v has $g = n^{2/3}d^{1/3}$ parents. One of them, u, has in turn $\gamma = n^{1/3}d^{-1/3}$ parents itself. Finally, there are $n' = n - g - \gamma - 2$ nodes that serve as additional children of the remaining $g-1$ parents of v; each parent chooses as children $\Delta = n^{1/3}d^{2/3}$ of those nodes. Note that G has $m = \Theta(g\Delta) = \Theta(nd)$ arcs. One can check that $P(v) = \Theta(\gamma/n) = \Theta(g/\Delta n)$, and that by reversing the arcs between u and its parents, $P(v)$ changes by a multiplicative factor $\Theta(\gamma)$. To estimate $P(v)$ one must decide the orientation of those arcs. It is easy to see that doing so with non-vanishing probability requires $\Theta(g)$ PARENT() queries and/or $\Theta(n/\gamma)$ JUMP() queries. However, $\Theta(g) = \Theta(n/\gamma) = \Theta(m^{2/3}d^{-1/3}) = \Theta(m^{1/2}\Delta^{1/2}d^{-1/2})$.

V. Adaptation to Heat Kernel

Porting our results to the Heat Kernel centrality requires lengthy adaptations. The crucial point is that, while PageRank weights paths of length τ by a factor α^τ, Heat Kernel weights them by a factor $\alpha^\tau/\tau!$ (see (2)). This means that, unlike PageRank, the probability of following a given path $\pi = (v_0, \ldots, v_t)$ in G does *not* equal the product of the probabilities of following the two sub-paths $\pi_1 = (v_0, \ldots, v_\tau)$ and $\pi_2 = (v_\tau, \ldots, v_t)$. This implies, for example, that we cannot rewrite $P(v)$ directly in terms of the scores $P(u)$ of the ancestors of v, as in (3). Instead, we must break $P(v)$ over the paths of length τ for all τ; that is, we write $P(v) = \sum_{\tau \ge 0} P_\tau(v)$. We then consider each $P_\tau(v)$ separately in most parts of our construction (and we can ignore all $\tau > a\lg(n/\epsilon)$ for some sufficiently large a, since for those lengths the score contribution is smaller than $\epsilon P(v)$). All the techniques and proofs can then be adapted, obtaining essentially the same bounds of PageRank (see [42], Appendix A.12).

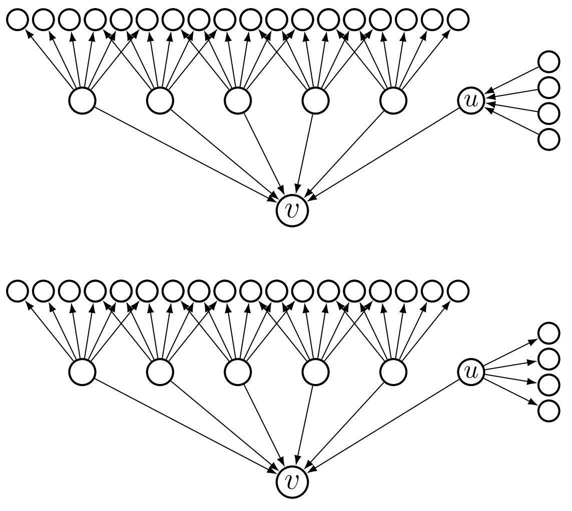

Figure 1. The generic graph G of the lower bound proof.

VI. Adaptation to the model of [1]

One can easily adapt our upper and lower bounds to the model of [1], which provides both JUMP() and a query NEIGH(u) that returns all incoming and outgoing arcs of u in one shot (see Section I). For the full proof see [42], Appendix A.13.

Let us start with the upper bounds. For an intuition, suppose first that NEIGH(u) returns, in addition to the parents and the children of u, the outdegree of each parent of u. In this case, expanding u requires just one query call: NEIGH(u). Therefore we can build $p_k(v)$ with $O(k)$ queries (see Section III-D), and since the sampling phase takes $O(\ell)$ queries, we can minimize the overall number of queries by setting $\ell = k$. Since by the bounds of Section III-E we need $k\ell = \Theta(n\epsilon^{-2}\ln(1/\delta))$, we finally obtain $k = \ell = O(n^{1/2}\epsilon^{-1}\ln(1/\delta)^{1/2}) = O(n^{1/2})$.

Unfortunately, NEIGH(u) does not return the outdegrees of the parents of u. Nonetheless, we can work with an approximation of these outdegrees, obtained beforehand via a global sketching phase. To this end we observe that, for any $u \in G$, the probability that a node z returned by JUMP() is a child of u is exactly $out(u)/n$; we can thus estimate $out(u)$ by repeated sampling. For each $u \in G$ let then $\widehat{out}(u)$ be an estimate of $out(u)$. For any pair of nodes $u, u' \in G$ with positive outdegrees let $r(u,u') = \frac{out(u)}{out(u')} / \frac{\widehat{out}(u)}{\widehat{out}(u')} \ge 1$ (if $r(u,u') < 1$ then just switch u and u'). Now let γ be the maximum value of $r(u,u')$ over all pairs of nodes having positive outdegree. We show that, with k queries, one can obtain estimates that make $\gamma \le n\ln(n)/k$ with probability sufficiently large.

We then build $p_k(v)$ as usual (Section III-D), but for all nodes we encounter, we use $\widehat{out}(u)$ in place of $out(u)$. This affects the choice of the weights $\beta_1, \beta_2, \ldots$ and of the nodes $u_1, u_2, \ldots$, and thus affects the coefficients of $p_k(v)$. In fact, $p_k(v)$ is no longer a perfect weighted estimator; but, crucially, we can show its coefficients satisfy:

$$\begin{aligned} c_k(u_i) &\ge c_k(u_{i'})/\gamma \quad && \forall i, i' \in \{1, \ldots, k\} \\ c_k(u_i) &\ge c_k(u)/\gamma && \forall i \in \{1, \ldots, k\}, \forall u \in F(G_k) \end{aligned}$$

It is easy to see that this implies a loss of a factor γ in the expectation of the random variable $c^{-1}(p_k(v) - c_k)$, which controls the concentration bounds (see Section III-E). In the end, we obtain:

$$\Pr\Big[\frac{|p_k^\ell(v) - P(v)|}{P(v)} > \epsilon\Big] < 2e^{-\epsilon^2(1-\alpha)k\ell/3\gamma n} \qquad (16)$$

In other words we are loosing a factor γ at the exponent of the concentration bounds. By plugging the bound $\gamma \leq n\ln(n)/k$ at the exponent of (16) and setting again $k = \ell$, we obtain a total of $O\big(n^{2/3}\ln(n)^{1/3}\ln(1/\delta)^{1/3}\epsilon^{-2/3}\big) = \tilde{O}\big(n^{2/3}\big)$ queries.

For the lower bounds, we can adapt the graph of Section IV as follows. We set the number of parents of v to $g = \Theta(n^{2/3})$; the number of children of each parent to $\Delta = \Theta(n^{1/3})$, ensuring distinct parents have no children in common; and γ to $\Theta(n^{1/3})$. One can easily prove that deciding the orientation of the arcs of the rightmost parent requires in expectation either $\Omega(n/\gamma) = \Omega(n^{2/3})$ JUMP() calls or $\Omega(g) = \Omega(n^{2/3})$ NEIGH() calls.

VII. Conclusions

We have shown how one can estimate the PageRank and Heat Kernel centrality of any given node in a graph, with rigorous approximation guarantees and fully sublinear query and computational cost. The key ingredients of this result are a novel low-variance estimator based on the exploration of a subgraph surrounding the target node, and a blacklisting scheme to avoid exploring nodes of large indegree. We believe our techniques may find application in other local graph computation problems, such as estimating the stationary probabilities of Markov chains or the solution vectors of linear systems; or in faster algorithms for approximating the entire centrality vector.

Acknowledgments

Marco Bressan is supported in part by the SIR Grant RBSI14Q743. Enoch Peserico is supported in part by Università degli Studi di Padova under Project CAEPAE.

References

[1] M. Brautbar and M. Kearns, "Local algorithms for finding interesting individuals in large networks," in *Proc. of ICS*, 2010, pp. 188–199.

[2] R. Andersen, F. Chung, and K. Lang, "Local graph partitioning using PageRank vectors," in *Proc. of IEEE FOCS*, 2006, pp. 475–486.

[3] D. A. Spielman and S.-H. Teng, "A local clustering algorithm for massive graphs and its application to nearly linear time graph partitioning," *SIAM J. Comput.*, vol. 42, no. 1, pp. 1–26, 2013.

[4] K. Fountoulakis, D. F. Gleich, and M. W. Mahoney, "An optimization approach to locally-biased graph algorithms," *Proceedings of the IEEE*, vol. 105, no. 2, pp. 256–272, 2017.

[5] S. Brin and L. Page, "The anatomy of a large scale hypertextual Web search engine," in *Proc. of WWW*, 1998.

[6] D. Saez-Trumper, G. Comarela, V. Almeida, R. Baeza-Yates, and F. Benevenuto, "Finding trendsetters in information networks," in *Proc. of ACM KDD*, 2012, pp. 1014–1022.

[7] Z. Gyöngyi, H. Garcia-Molina, and J. Pedersen, "Combating web spam with TrustRank," in *Proc. of VLDB*, 2004, pp. 576–587.

[8] P. Gupta, A. Goel, J. Lin, A. Sharma, D. Wang, and R. Zadeh, "WTF: The Who To Follow service at Twitter," in *Proc. of WWW*, 2013, pp. 505–514.

[9] D. F. Gleich, "PageRank beyond the web," *SIAM Review*, vol. 57, no. 3, pp. 321–363, 2015.

[10] F. Chung, "A brief survey of PageRank algorithms," *IEEE Transactions on Network Science and Engineering*, vol. 1, no. 1, pp. 38–42, 2014.

[11] X. Wu, V. Kumar, J. Ross Quinlan, J. Ghosh, Q. Yang, H. Motoda, G. J. McLachlan, A. Ng, B. Liu, P. S. Yu, Z.-H. Zhou, M. Steinbach, D. J. Hand, and D. Steinberg, "Top 10 algorithms in data mining," *Knowl. Inf. Syst.*, vol. 14, pp. 1–37, 2007.

[12] F. Chung, "The heat kernel as the pagerank of a graph," *Proceedings of the National Academy of Sciences*, vol. 104, no. 50, pp. 19 735–19 740, 2007.

[13] E. Estrada and J. A. Rodriguez-Velazquez, "Subgraph centrality in complex networks," *Physical Review E*, vol. 71, no. 5, 2005.

[14] E. Estrada and D. J. Higham, "Network properties revealed through matrix functions," *SIAM Review*, vol. 52, no. 4, pp. 696–714, 2010.

[15] F. Chung and O. Simpson, "Solving local linear systems with boundary conditions using heat kernel PageRank," *Internet Mathematics*, vol. 11, no. 4-5, pp. 449–471, 2015.

[16] F. Chung, "A local graph partitioning algorithm using heat kernel pagerank," in *Proc. of WAW*, 2009, pp. 62–75.

[17] ——, "A local graph partitioning algorithm using heat kernel pagerank," *Internet Mathematics*, vol. 6, no. 3, pp. 315–330, 2009.

[18] K. Kloster and D. F. Gleich, "Heat kernel based community detection," in *Proc. of ACM KDD*, 2014, pp. 1386–1395.

[19] F. Chung and O. Simpson, "Computing heat kernel pagerank and a local clustering algorithm," *European Journal of Combinatorics*, vol. 68, pp. 96–119, 2018.

[20] Y.-Y. Chen, Q. Gan, and T. Suel, "Local methods for estimating PageRank values," in *Proc. of ACM CIKM*, 2004, pp. 381–389.

[21] D. Gleich and M. Polito, "Approximating personalized PageRank with minimal use of web graph data," *Internet Mathematics*, vol. 3, no. 3, pp. 257–294, 2007.

[22] R. Andersen, C. Borgs, J. Chayes, J. Hopcroft, V. Mirrokni, and S.-H. Teng, "Local computation of PageRank contributions," *Internet Mathematics*, vol. 5, no. 1–2, pp. 23–45, 2008.

[23] Z. Bar-Yossef and L.-T. Mashiach, "Local approximation of PageRank and reverse PageRank," in *Proc. of ACM SIGIR*, 2008, pp. 865–866.

[24] ——, "Local approximation of PageRank and reverse PageRank," in *Proc. of ACM CIKM*, 2008, pp. 279–288.

[25] M. Bressan and L. Pretto, "Local computation of PageRank: the ranking side," in *Proc. of ACM CIKM*, 2011, pp. 631–640.

[26] C. Borgs, M. Brautbar, J. T. Chayes, and S.-H. Teng, "A sublinear time algorithm for PageRank computations," in *Proc. of WAW*, 2012, pp. 41–53.

[27] ——, "Multi-scale matrix sampling and sublinear-time PageRank computation," *CoRR*, vol. abs/1202.2771, 2012.

[28] P. A. Lofgren, S. Banerjee, A. Goel, and C. Seshadhri, "FAST-PPR: Scaling personalized PageRank estimation for large graphs," in *Proc. of ACM KDD*, 2014, pp. 1436–1445.

[29] P. Lofgren, S. Banerjee, and A. Goel, "Bidirectional PageRank estimation: From average-case to worst-case," in *Proc. of WAW*, 2015, pp. 164–176.

[30] ——, "Personalized PageRank estimation and search: A bidirectional approach," in *Proc. of ACM WSDM*, 2016, pp. 163–172.

[31] K. Avrachenkov, N. Litvak, D. Nemirovsky, and N. Osipova, "Monte Carlo methods in PageRank computation: When one iteration is sufficient," *SIAM Journal on Numerical Analysis*, vol. 45, no. 2, pp. 890–904, 2007.

[32] C. Borgs, M. Brautbar, J. T. Chayes, and S.-H. Teng, "Sublinear time algorithm for PageRank computations and related applications," *CoRR*, vol. abs/1202.2771, 2012.

[33] M. Bressan, E. Peserico, and L. Pretto, "The power of local information in PageRank," in *Proc. of WWW (Companion Volume)*, 2013, pp. 179–180.

[34] K. Kloster and D. F. Gleich, "A nearly-sublinear method for approximating a column of the matrix exponential for matrices from large, sparse networks," in *Proc. of WAW*, 2013, pp. 68–79.

[35] D. F. Gleich and K. Kloster, "Sublinear column-wise actions of the matrix exponential on social networks," *Internet Mathematics*, vol. 11, no. 4-5, pp. 352–384, 2015.

[36] O. Goldreich, S. Goldwasser, and D. Ron, "Property testing and its connection to learning and approximation," *J. ACM*, vol. 45, no. 4, pp. 653–750, Jul. 1998.

[37] O. Goldreich and D. Ron, "Property testing in bounded degree graphs," *Algorithmica*, vol. 32, no. 2, pp. 302–343, 2002.

[38] B. Bahmani, R. Kumar, M. Mahdian, and E. Upfal, "PageRank on an evolving graph," in *Proc. of ACM KDD*, 2012, pp. 24–32.

[39] A. Dasgupta, R. Kumar, and T. Sarlos, "On estimating the average degree," in *Proc. of WWW*, 2014, pp. 795–806.

[40] B. Lucier, J. Oren, and Y. Singer, "Influence at scale: Distributed computation of complex contagion in networks," in *Proc. of ACM KDD*, 2015, pp. 735–744.

[41] F. Chierichetti, A. Dasgupta, R. Kumar, S. Lattanzi, and T. Sarlós, "On sampling nodes in a network," in *Proc. of WWW*, 2016, pp. 471–481.

[42] M. Bressan, E. Peserico, and L. Pretto, "Sublinear algorithms for local graph centrality estimation," *CoRR*, vol. abs/1404.1864, 2018.

[43] ——, "Simple set cardinality estimation through random sampling," *CoRR*, vol. abs/1512.07901, 2015.

[44] D. Fogaras, B. Rácz, K. Csalogány, and T. Sarlós, "Towards scaling fully personalized PageRank: Algorithms, lower bounds, and experiments," *Internet Mathematics*, vol. 2, no. 3, pp. 333–358, 2005.

[45] R. Andersen, C. Borgs, J. Chayes, J. Hopcroft, V. S. Mirronki, and S.-H. Teng, "Local computation of PageRank contributions," in *Proc. of WAW*, 2007, pp. 150–165.

[46] M. Bressan, E. Peserico, and L. Pretto, "The power of local information in PageRank," *CoRR*, vol. abs/1604.00202, 2016.

[47] S. Banerjee and P. Lofgren, "Fast bidirectional probability estimation in Markov models," in *Proc. of NIPS*, 2015, pp. 1423–1431.

[48] A. H. Al-Mohy and N. J. Higham, "Computing the action of the matrix exponential, with an application to exponential integrators," *SIAM Journal on Scientific Computing*, vol. 33, no. 2, pp. 488–511, 2011.

[49] L. Orecchia, S. Sachdeva, and N. K. Vishnoi, "Approximating the exponential, the Lanczos method and an $\tilde{O}(m)$-time spectral algorithm for balanced separator," in *Proc. of ACM STOC*, 2012, pp. 1141–1160.

[50] C. E. Lee, A. Ozdaglar, and D. Shah, "Computing the stationary distribution locally," in *Proc. of NIPS*, 2013, pp. 1376–1384.

[51] M. Bressan, E. Peserico, and L. Pretto, "On Approximating the Stationary Distribution of Time-reversible Markov Chains," in *Proc. of STACS*, 2018, pp. 18:1–18:14.

[52] C. E. Lee, A. Ozdaglar, and D. Shah, "Asynchronous approximation of a single component of the solution to a linear system," *CoRR*, vol. abs/1411.2647, 2014.

[53] N. Shyamkumar, S. Banerjee, and P. Lofgren, "Sublinear estimation of a single element in sparse linear systems," in *Proc. of Allerton*, 2016, pp. 856–860.

[54] L. Katz, "A new status index derived from sociometric analysis," *Psychometrika*, vol. 28, pp. 39–43, 1953.