Approximating PageRank locally with sublinear query complexity

Marco Bressan[1], Enoch Peserico[2], and Luca Pretto[2]

[1]Università di Roma La Sapienza, Roma, Italy
[2]Università degli Studi di Padova, Padova, Italy

July 13, 2016

Abstract

The problem of approximating the centrality of a node with minimal information about the rest of the graph has attracted considerable attention in the last decade; but its central question, whether it is in general necessary to explore a non-vanishing fraction of the graph, is still open in many prominent cases. In this paper we present the first algorithm for approximating the PageRank score of a given node in a graph by exploring only a vanishing fraction of the graph's topology. More formally, given any one node in any n-node graph, with probability $1-\delta$ our algorithm produces a multiplicative $(1\pm\epsilon)$ approximation of its PageRank score using at most $O\big(n^{\frac{2}{3}}\sqrt[3]{\log(n/\delta)}\,\ln(1/\delta)^{\frac{1}{3}}\epsilon^{-\frac{2}{3}}\big) = \tilde{O}(n^{\frac{2}{3}})$ graph exploration queries, where with a single query one can pick a node at random in the graph or retrieve the list of neighbours of a given node. We show that the algorithm is essentially optimal by proving a $\Omega(n^{\frac{2}{3}})$ lower bound.

Our result is based on a novel technique for constructing a low-variance estimator of the score of a target node through a progressive exploration of the graph. This technique allows one to approximate the score with $O(n^{\frac{1}{2}}\gamma^{\frac{1}{2}})$ queries, assuming one knows beforehand a γ-approximation of the relative ratios of the graph's outdegrees. We then give an outdegree sketching scheme that ensures $\gamma=\tilde{O}(\frac{n}{k})$ with high probability using k queries, which yields the bound above for $k=\tilde{O}(n^{\frac{2}{3}})$. Our techniques are rather general, and may help in designing low-query-complexity algorithms for other local approximation problems.

1 Introduction

The massive size and evolving nature of modern networks makes it increasingly difficult to use network analysis algorithms that need to process the whole input graph. This has led to the development of alternative, efficient paradigms to compute an approximate solution by examining a minimal fraction of the graph, for problems spanning from estimating the average node degree [17] to finding a well-connected cluster around a given node [35, 37]. In this context, a natural problem is computing the *centrality* of a given node in the graph, which is by now considered a key primitive in large network analysis applications. Not surprisingly then, much work has focused on PageRank [14], probably the most celebrated graph centrality measure; indeed, the problem of computing the PageRank score of a given node by exploring a small fraction of the graph was posed over a decade ago [15], and has attracted considerable attention since then [20, 1, 7, 6, 13, 9, 8, 26, 30, 27, 28]. Yet, the basic question of whether exploring a non-vanishing fraction of a graph is in general necessary to approximate such a score remained open until now (only for specific graphs and/or nodes was a solution known).

In this paper we present the first algorithm to approximate the PageRank score of a node exploring only a vanishingly small portion of its graph regardless of the graph's topology or the specific node under investigation. Our techniques are of independent interest, and consist in combining random walks with a local exploration of the graph guided by a global sketching of the graph's outdegrees.

The paper is organized as follows. This introduction briefly reviews PageRank and its local approximation (Subsection 1.1), and summarizes the state of the art (Subsection 1.2). Section 2 outlines the main results of the paper. Section 3 contains an in-depth walkthrough of the main ideas and techniques used to prove our theorems. All the details omitted to ease the reader's burden can be found in the appendix.

1.1 Local approximation of PageRank

The PageRank score of a node u in a graph G of n nodes $u_1, \dots u_n$ can be easily defined in terms of an abstract *surfer*, who navigates G starting from an arbitrary node. At each timestep, with some positive probability α the surfer moves to a child of the current node chosen uniformly at random, or to a node chosen uniformly at random if the current node is *dangling*, i.e. has no children. Otherwise, with positive probability $1-\alpha$ the surfer performs a *random jump* leading to a node chosen uniformly at random; the parameter α is called the *damping factor*. The PageRank score $P(u_i)$ of u_i is simply the stationary probability of the surfer being on u_i. The row vector $\mathbf{p}$ whose i^{th} component is $P(u_i)$ is then the unique solution of:

$$\mathbf{p} = \alpha \mathbf{p}\mathbf{M} + \frac{(1-\alpha)}{n}\mathbf{1} \tag{1}$$

where $\mathbf{1}$ is the n-dimensional row vector whose components all equal 1 and $\mathbf{M}$ is an n by n matrix whose generic element $M_{i,j}$ equals n^{-1} if u_i has no children, and equals m^{-1} or 0 if u_i has $m > 0$ children and u_j respectively is or is not one of them. Thus, $\frac{1-\alpha}{n} \leq P(u_i) \leq 1$.

In many cases of practical interest only the PageRank score of a single node or of a very small set of nodes is needed [32, 24, 21, 36, 19]. Computing those scores from Equation 1 using standard linear algebra tools requires retrieving and manipulating the entire adjacency matrix of the graph; this can be prohibitively expensive or even impossible for graphs, such as the web and many social networks, which are massive and/or rapidly evolving or whose access is heavily limited. These practical motivations lead to the problem of obtaining a local approximation of PageRank: formally speaking, approximate the PageRank score of a target node within a factor $(1 \pm \epsilon)$ with probability at least $1-\delta$ by exploring only $o(n)$ nodes of the graph, for some small $\epsilon, \delta > 0$. Let us now describe more formally how an algorithm can explore the graph.

The precise way an algorithm can access the graph is defined by a set of available *exploration queries*; each query receives an input (possibly empty) and reveals a (typically tiny) portion

of the graph in output. Exploration queries permit an incremental exploration of the graph, and the cost or *query complexity* of an algorithm is the number of queries it performs. We adopt the three standard exploration queries used in local PageRank computation. The first, *Neighbourhood(u)* [6, 4], returns a list of all arcs entering and/or leaving a node u. The second, *RandomChild(u)* [10, 9], also known as *crawl*, returns a child of u chosen uniformly at random, or the empty set if u is childless. Finally, *RandomNode()* [10, 9], also known as *jump*, returns a random node of the graph. These queries are standard primitives to access large graphs in many contexts [17, 31, 16]. In line with previous work, we assume the algorithm possesses no initial information about the graph.

The above three queries, and the others employed in the literature, can be divided in two classes: *local queries*, that reveal the graph's local structure around a given node, that allow one to discover nodes previously unknown. Both classes are subsumed by the notion of *natural exploration queries* [12] which, essentially, never disclose more that one row and one column of the graph's adjacency matrix. Our lower bounds hold for all natural queries; also, algorithms based on natural queries can be ported to other graph access models via straightforward adaptations, as we do for our upper bounds (see Section 2).

1.2 State of the art

The problem of approximating PageRank locally with minimal information about the graph was first introduced a decade ago in [15] under a *Neighbourhood(·)*-based graph access model, together with some heuristic exploration strategies which yielded encouraging experimental results. Subsequently, [1] gave a *Neighbourhood(·)*-based algorithm to find the nodes that contribute the most to the score of a target node; it can also be used as a local PageRank approximation algorithm, but in the worst case it needs to perform $\Omega(n)$ queries. These results were then countered by lower bounds. First, [6] proved that deterministic $(1 \pm \epsilon)$-approximation algorithms need $\Omega(n)$ *Neighbourhood(·)* queries and that Las Vegas and Monte Carlo ones need $\Omega(\sqrt{n})$. Later, [13] showed these bounds hold also for the simpler ranking problem (determine which of two nodes has the higher score, without necessarily computing it). Finally, [12] strengthened these bounds to $n(1-o(1))$ for any deterministic or Las Vegas algorithm employing any combination of natural queries, and proved bounds of $\Omega(n)$ for any Monte Carlo algorithm using only local queries.

At the same time, several works addressed the related problem of approximating the PageRank scores of the top-ranking nodes in the graph. In particular, [18, 3, 9, 8] developed a simple but powerful technique to sample nodes from a graph with probability proportional to their PageRank score using $O(1)$ jump-and-crawl (i.e. *RandomNode()* and *RandomChild(·)*) queries per sample. Using this technique one can obtain a $(1 \pm \epsilon)$ approximation of the PageRank score $P(v)$ of a node v with $\tilde{O}(1/P(v))$ jump-and-crawl queries; and [9, 8] also proved that $\Omega(1/P(v))$ queries are in general necessary if one is restricted to jump-and-crawl queries. Since at most a vanishing fraction of all nodes in a graph can have PageRank score asymptotically larger than $1/n$, breaking the $\Omega(n)$ query complexity barrier with only jump-and-crawl queries is in general impossible.

Finally, some recent papers [30, 27, 5, 28] addressed the problem of efficiently approximating the Personalized PageRank (PPR) [14] of a target node. Their main result is an algorithm which approximates the PPR score of a node within a factor $(1 \pm \epsilon)$ whenever the score is larger than a given δ; for a randomly chosen node the algorithm has expected running time $\tilde{O}(\sqrt{d/\delta})$, where d is the average degree of the graph. This algorithm can be ported to the computational model of the present paper, but in this context does not guarantee worst-case $o(n)$ query complexity, requiring $\Theta(n)$ natural queries in general even if granted knowledge of the outdegrees of graph.

2 Our contribution

We present the first algorithm that breaks the linear query complexity barrier of local PageRank approximation for any given node in any given graph. Our main result is the following.

Theorem 1. $O\big(n^{\frac{2}{3}}\sqrt[3]{\log(n/\delta)}\,\ln(1/\delta)^{\frac{1}{3}}\epsilon^{-\frac{2}{3}}\big)$ RandomNode() *and* Neighbourhood$(\cdot)$ *queries are sufficient to obtain with probability* $(1-\delta)$ *a* $(1\pm\epsilon)$*-approximation of the PageRank score of any node in any* n*-node graph.*

This bound is essentially optimal: we prove that any algorithm employing only natural queries that computes with sufficiently high probability a $(1\pm\epsilon)$-approximation of the PageRank score of a node must in general make $\Omega(n^{\frac{2}{3}})$ queries. The main tools behind the proof of Theorem 1 are a very general estimator of the PageRank score of a node based on the topology of an explored subgraph, and an iterative subgraph exploration technique that minimizes the variance of that estimator with an optimal number of queries. These tools can be abstracted and may be of use in other local graph approximation problems; Appendix A.11 briefly discusses two cases, Katz's centrality and steady-state probabilities of ergodic Markov chains, that we leave as future work.

Our second result, arising along the proof of Theorem 1, is an alternative upper bound parameterized on the knowledge of the outdegrees of the graph.

Theorem 2. *Let out(u) denote the outdegree of* u*. If for any pair* u, u' *of nodes with positive outdegree one has an estimate of* $out(u')/out(u)$ *that is within a multiplicative factor* γ *of the actual value, then the bound of Theorem 1 becomes* $O\big(n^{\frac{1}{2}}\gamma^{\frac{1}{2}}\sqrt{\ln(1/\delta)}\frac{1}{\epsilon}\big)$.

An immediate implication of this theorem is that estimating the PageRank score of a node in graphs whose outdegree is known to be bounded by γ can be done using just $O(n^{\frac{1}{2}}\gamma^{\frac{1}{2}})$ queries, which in many web and social network models (see e.g. [25]) is better than the general bound of Theorem 1. The bound is also tight in this sense: we show that $\Omega(n^{\frac{1}{2}}\gamma^{\frac{1}{2}})$ queries may be necessary if the graph's outdegree is bounded by γ.

We complement Theorems 1 and 2 with a slightly complex but very general lower bound on the number of natural queries needed to approximate the PageRank score of a node. In fact, we prove the bound for an easier *ranking* problem which asks to correctly determine which of two given nodes in a graph has the higher PageRank score, provided the two scores are not too close. The bound on approximating the score immediately follows, since one can rank the two nodes correctly if one can obtain $(1\pm\epsilon)$-approximations of their scores for a small enough ϵ.

Theorem 3. *Choose any real functions* $f(n)$, $\gamma(n)$ *with* $\frac{1}{n} \le f(n) \le 1$ *and* $\gamma(n) \ge 1$*, any constant* $\eta > 0$*, and any (Monte Carlo) algorithm using only natural queries. There exists a graph of arbitrarily large size* n *and maximum outdegree* $O(\gamma(n))$ *containing two nodes* u *and* v *with* $P(u), P(v)$ *in* $\Theta(f(n))$ *and* $P(u) > (1+\eta)P(v)$ *such that the algorithm must use* $\Omega\big(\min\big(n^{\frac{2}{3}}, n^{\frac{1}{2}}\gamma(n)^{\frac{1}{2}}, \frac{1}{f(n)}\big)\big)$ *queries in expectation to have probability* $\frac{1}{2}+\Omega(1)$ *of correctly determining that* $P(u) > P(v)$.

Theorem 3 combines three bounds. The first two, $\Omega(n^{\frac{2}{3}})$ and $\Omega(n^{\frac{1}{2}}\gamma(n)^{\frac{1}{2}})$, are the first non-trivial bounds applying to algorithms based on natural queries, including all existing algorithms for the problem; the third, $\Omega(\frac{1}{f(n)})$, is known for the jump-and-crawl model. Note that this is strictly stronger than the three bounds taken separately. Due to space limitations, the proof of Theorem 3 is given in Appendix A.12.

It is worth noting that Theorems 1 and 2 yield immediate counterparts in the graph access models of [22, 23], used in property testing and related fields, where each query allows inspecting only a single arc. Indeed, in their bounded-degree model (where the degree of the graph is $O(1)$) and dense graph model (where the graph has $\Theta(n^2)$ arcs) one can emulate *Neighbourhood(u)* with respectively $O(1)$ and $O(n)$ of their queries – in both cases, the query complexity of our algorithm remains sublinear in the total number of *arcs*.

3 Upper Bounds

This section provides a detailed walkthrough of the main ideas and techniques behind the proofs of Theorems 1 and 2. All details can be found in the appendix.

At a high level, our results stem from two main ideas. The first, originally delineated in [18, 3], is a simple random walk technique that allows one to sample a node with probability equal to its PageRank score; this technique alone requires $\Theta(n)$ samples and queries to estimate $P(v)$ in the worst case. The second, developed simultaneously in [11] and [29], consists in building around v an "antenna" subgraph that gets hit by many more walks than the few ones hitting v directly, which can reduce the number of samples and queries required. Obtaining a *worst-case* sublinear query complexity through these techniques is however nontrivial: the challenge is to build an antenna subgraph of only $o(n)$ nodes, yet be able to derive from it enough information to estimate $P(v)$ with just $o(n)$ random walk samples, employing only natural queries and without any prior information about the graph.

Let us outline the key steps of the proof. We begin by developing in Subsection 3.1 a general "diffuse" unbiased estimator of $P(v)$ that is a weighted sum of random variables, each one tied to a node in a subgraph surrounding v and indicating if it gets hit by a random walk. In Subsection 3.2 we derive concentration bounds showing that the variance of our estimator, and thus the number of queries required to estimate $P(v)$, is controlled by the relative imbalance between the coefficients of the weighted sum. We then present an iterative graph exploration scheme that, exploiting the general formulation of our estimator, builds one with perfectly balanced coefficients using an optimal number of queries, assuming the graph's outdegrees are known beforehand – in which case we obtain a $O(n^{\frac{1}{2}})$ query complexity upper bound. Subsection 3.3 shows this is a special case of a more general scenario: if one disposes beforehand of estimates that distort by a factor at most γ the true ratio between any two outdegrees, then $P(v)$ can be estimated with just $O(n^{\frac{1}{2}}\gamma^{\frac{1}{2}})$ queries, proving Theorem 2. Finally, in Section 3.4 we provide a simple outdegree approximation scheme that yields $\gamma = O(n\log(n)/k)$ with high probability using k queries, which for $k = \tilde{O}(n^{\frac{2}{3}})$ gives the $\tilde{O}(n^{\frac{2}{3}})$ query complexity upper bound of Theorem 1.

Let us now delve into the proof.

Remark. Our algorithm, and the proof of the corresponding bounds below, require knowledge of the number n of nodes in the graph, which is initially unknown. One can estimate n by counting how many *RandomNode()* calls are needed to make the number of repeats among the sampled nodes larger than a given threshold; this intuition is formalized in Appendix A.2 which shows how, for any fixed $\epsilon, \delta > 0$, with probability larger than $1-\delta$ one can approximate n within a factor $(1 \pm \epsilon)$ using $O\big(n^{\frac{1}{2}}\sqrt{\ln(1/\delta)}\frac{1}{\epsilon}\big)$ *RandomNode()* calls. Such an approximation allows our algorithm to work with the same error guarantees as if n were known, while maintaining the same asymptotic query complexity. Therefore, from now on we shall assume n is known.

3.1 A diffuse local PageRank estimator

The first cornerstone of our algorithm, presented in this section, is a "diffuse" estimator of $P(v)$ which, in a nutshell, employs a subgraph around v as an "antenna" that collects the information obtained by sampling nodes through PageRank random walks. This estimator is based on three ingredients, introduced in the following subsections: a procedure for sampling nodes with probability equal to their PageRank score, a "diffuse" formulation of $P(v)$ as a linear combination of the PageRank scores of nodes surrounding v, and a node weighting scheme that helps controlling the variance of the estimator by adding degrees of freedom.

Sampling nodes with probability equal to their PageRank score. The first ingredient of our estimator is *SampleNode()*, a simple node-sampling routine originally introduced in [18, 3] and which is formally defined in Appendix A.5. *SampleNode()* emulates PageRank's random walk using *RandomNode()* and *RandomChild(·)* queries, starting from a node chosen uniformly at random and returning the last node visited before a random jump. As stated in the following lemma, proved in Appendix A.6, this routine allows one to sample nodes with probability equal to their PageRank score:

Lemma 4. SampleNode() *returns node u with probability $P(u)$.*

It is immediate to see from the pseudocode in Appendix A.5 that one call to *SampleNode()* makes $O(1)$ queries in expectation. Also, by standard concentration bounds, the probability that $O(\ell)$ queries are not sufficient to complete ℓ calls is exponentially small in ℓ (see Appendix A.7); in our analysis we will therefore equivalently use the number of *SampleNode()* calls in place of the number of queries required to perform those calls. Finally, note that we can replace each call to *RandomChild(·)* with a call to *Neighbourhood(·)* followed by picking one at random of the outgoing neighbours returned.

A diffuse formulation of PageRank. To state our diffuse formulation we need to introduce the notion of *PageRank conductance* $\mho_{G'}(u, v)$ of a subgraph G' of G from a node u to a node v. Informally, $\mho_{G'}(u, v)$ is the expected number of times a PageRank random walk starting in u goes through v before leaving G' or taking a random jump; a complete formal definition is given in Appendix A.3.

A second notion we need, and that will also be used in the rest of the paper, is that of *frontier* of an induced subgraph $\bar{G}$ of G – informally, the set of arcs leading from nodes of $G \setminus \bar{G}$ to nodes of $\bar{G}$. Formally, given an induced subgraph $\bar{G} = (\bar{V}, \bar{E})$ of $G = (V, E)$, its frontier is $F(\bar{G}) = \{(u, w) \in E : u \notin \bar{V}, w \in \bar{V}\}$.

The key idea is then the following. Consider an induced subgraph $\bar{G}$ of G containing v, and recall (see Subsection 1.1) that $P(v)$ is the stationary probability that the PageRank random walk is in v. For the random walk to reach v, one of two disjoint events must occur since the last random jump takes place. In the first, the jump takes the walk to some node w in $\bar{G}$, and then the walk reaches v by following a sequence of arcs without leaving $\bar{G}$. In the second, at some point the walk is outside $\bar{G}$; thus it must first reach some node u having an outgoing arc (u, w) in $F(\bar{G})$, then follow (u, w) to enter $\bar{G}$, and finally reach v by following a sequence of arcs without leaving $\bar{G}$ anymore. The probabilities of these events, whose sum is $P(v)$, can be expressed as a function of the PageRank score of the nodes just outside $\bar{G}$, and of their outdegrees, by using the notions of PageRank conductance and subgraph frontier introduced above. Let indeed $out(u)$ denote the outdegree of u in the supergraph G; the following lemma, proved in Appendix A.4, gives our diffuse formulation of $P(v)$.

Lemma 5. *Given a graph G without dangling nodes, a node $v \in G$, and an induced subgraph $\bar{G}$ of G containing v, the PageRank score of v can be written as:*

$$P(v) = \sum_{w \in \bar{G}} \frac{1-\alpha}{n} \mho_{\bar{G}}(w, v) \; + \sum_{(u,w) \in F(\bar{G})} P(u) \cdot \frac{\alpha}{out(u)} \mho_{\bar{G}}(w, v) \tag{2}$$

The assumption that G is free from dangling nodes can be lifted without impacting on our final results, and makes the discussion much lighter. For these reasons, from now on we assume G does not contain dangling nodes. Appendix A.10 describes the simple modifications one needs to make in order to extend the discussion to the general case.

A first diffuse estimator of $P(v)$. We begin by rewriting our diffuse formulation in a more convenient form. If we let $c_{\bar{G}} = \sum_{w \in \bar{G}} \frac{1-\alpha}{n} \mho_{\bar{G}}(w, v)$ and $c_{\bar{G}}(u) = \sum_{w:(u,w) \in F(\bar{G})} \alpha \, \mho_{\bar{G}}(w, v)$, then Equation 2 becomes:

$$P(v) = c_{\bar{G}} + \sum_{u:(u,w) \in F(\bar{G})} P(u) \cdot \frac{c_{\bar{G}}(u)}{out(u)} \tag{3}$$

Note that the summation is now over the nodes appearing on (i.e. having an outgoing arc in) $F(\bar{G})$ instead of over the arcs in $F(\bar{G})$. Let now χ_u be the indicator random variable of the event that a single invocation of *SampleNode()* returns node u. By Lemma 4 we have $\mathbb{E}[\chi_u] = P(u)$,

and therefore if in Equation 3 we replace $P(u)$ by χ_u we obtain, by linearity of expectation, a random variable which is an unbiased estimator of $P(v)$. We have just proved:

Lemma 6. *$P(v) = \mathbb{E}[p_{\bar{G}}(v)]$, where*

$$p_{\bar{G}}(v) = c_{\bar{G}} + \sum_{u:(u,w)\in F(\bar{G})} \chi_u \cdot \frac{c_{\bar{G}}(u)}{out(u)} \tag{4}$$

This is a first "diffuse" estimator of $P(v)$: a weighted sum of random variables, each indicating if a node appearing on the frontier of $\bar{G}$ is hit by *SampleNode()*. Taking a sample of $p_{\bar{G}}(v)$ requires just one invocation of *SampleNode()*, the same required to take a sample of the naive estimator χ_v; but the key advantage of $p_{\bar{G}}(v)$ is that, when $P(v)$ is small, *SampleNode()* is much more likely to return some of the nodes on the frontier of $\bar{G}$ rather than v itself, and thus the variance of $p_{\bar{G}}(v)$ can be much lower than that of χ_v.

The crucial point is now the coefficients $\frac{c_{\bar{G}}(u)}{out(u)}$ of the random variables χ_u. If those coefficients were all identical, then the weighted sum in Equation 4 would have the lowest possible variance, and $\mathbb{E}[p_{\bar{G}}(v)]$ could be estimated with as few samples of $p_{\bar{G}}(v)$ as possible. Unfortunately, those coefficients are determined by the topology of $\bar{G}$, which we know only after exploring it, and by the outdegrees of the nodes in its immediate neighbourhood, which might even be $\Theta(n)$. To overcome these obstacles, we need a more sophisticated estimator which gives control over the coefficients of the random variables χ_u.

Gaining a degree of freedom by weighting nodes along the exploration. Lemma 6 states that, for a single induced subgraph $\bar{G}$ containing v, the random variable $p_{\bar{G}}(v)$ has expectation $P(v)$. Therefore, given any $m+1$ induced subgraphs $G_0, \ldots, G_m$ containing v, a linear combination of the random variables $p_{G_0}(v), \ldots, p_{G_m}(v)$ through multipliers $\beta_0^m, \ldots, \beta_m^m$ with sum 1 also has expectation $P(v)$. In other words, the random variable

$$p_m(v) = \sum_{i=0}^{m} \beta_i^m p_{G_i}(v) = \sum_{i=0}^{m} \beta_i^m \left(c_{G_i} + \sum_{u:(u,w)\in F(G_i)} \chi_u \cdot \frac{c_{G_i}(u)}{out(u)} \right) \tag{5}$$

is also an unbiased estimator of $P(v)$ whenever $\sum_{i=0}^{m} \beta_i^m = 1$.

Consider then a sequence of increasingly larger induced subgraphs $G_0, \ldots, G_m$, each one containing v, obtained by progressively exploring G through a sequence of *Neighbourhood($\cdot$)* queries in the following way. Start by querying v; this reveals the subgraph G_0 induced by v, as well as its frontier $F(G_0)$. Then, for each $i = 1, \ldots, m$, query a node on $F(G_{i-1})$, obtaining the subgraph G_i induced by that node and all the nodes already in G_{i-1}, as well as its frontier $F(G_i)$. Denote by $u_1, \ldots, u_m$ the nodes that have been queried in sequence after v, and for each u in G let $j(u)$ be the smallest index i such that some arc (u, w) is in $F(G_i)$. One can easily check that Equation 5 can be restated as:

$$p_m(v) = \sum_{i=0}^{m} \beta_i^m c_{G_i} + \sum_{i=1}^{m} \chi_{u_i} \left(\sum_{j=j(u_i)}^{i-1} \beta_j^m \frac{c_{G_j}(u_i)}{out(u_i)} \right) + \sum_{u:(u,w)\in F(G_m)} \chi_u \left(\sum_{j=j(u)}^{m} \beta_j^m \frac{c_{G_j}(u)}{out(u)} \right) \tag{6}$$

Equation 6 can be read as follows. The first summation simply collects all the constant terms from Equation 5. The second summation gathers the random variables χ_{u_i} associated to nodes u_i that were on the frontier of some G_i, but have since been queried and are now in G_m. Each χ_{u_i} is weighted by a linear combination, determined by the multipliers $\beta_0^m, \ldots, \beta_m^m$, of the coefficients it appears with in $p_0(v), \ldots, p_m(v)$; since u_i is only on the frontiers of $G_{j(u_i)}, \ldots, G_{i-1}$, only the relative coefficients appear in the linear combination, all the others being zero. The third summation is similar to the second, but concerns only those nodes that are still on $F(G_m)$.

The random variable $p_m(v)$ is our diffuse estimator of $P(v)$. Unlike $p_{G_m}(v)$ (see Lemma 6),

$p_m(v)$ takes into account also the random variables χ_{u_i} associated to the nodes $u_1, \ldots, u_m$ of G_m. But the crucial point is that, in $p_m(v)$, those random variables are weighted by coefficients that depend not only on the topology of the explored subgraphs, but also on multipliers $\beta_0^m, \ldots, \beta_m^m$ that we can control; as we will see, this helps keeping $p_m(v)$ concentrated around its mean.

3.2 Building an efficient estimator

This section shows how the general formulation of $p_m(v)$ can be used to build an efficient estimator for $P(v)$. We first derive bounds on the probability that $p_m(v)$ falls far from $P(v)$, highlighting the key parameters that determine the approximation guarantees. We then present a technique to build an estimator $p_m(v)$ as concentrated as possible with as few queries as possible if one knows the degrees of the graph in advance.

Concentration bounds for $p_m(v)$. We start by rewriting Equation 6 more compactly. First, we let $c_m = \sum_{i=0}^{m} \beta_i^m c_{G_i}$. Also, let $c_m(u_i) = \sum_{j=j(u_i)}^{i-1} \beta_j^m \frac{c_{G_j}(u_i)}{out(u_i)}$ for all $i = 1, \ldots, m$, and let $c_m(u) = \sum_{j=j(u)}^{m} \beta_j^m \frac{c_{G_j}(u)}{out(u)}$ for any u having an outgoing arc in $F(G_m)$. This gives:

$$p_m(v) = c_m + \sum_{i=1}^{m} \chi_{u_i} \cdot c_m(u_i) + \sum_{u:(u,w)\in F(G_m)} \chi_u \cdot c_m(u) \tag{7}$$

Equation 7 will be a focal point of our analysis. We will call *inner summation* its first summation, and *frontier summation* its second summation; similarly, we will call *inner coefficients* the coefficients $c_m(u_i)$, and *frontier coefficients* the coefficients $c_m(u)$.

We now turn to the probability bounds. Instead of $p_m(v)$, which is tied to a single invocation of *SampleNode()*, we analyse a more general estimator $p_m^\ell(v)$ which is just the average of $p_m(v)$ over ℓ independent *SampleNode()* invocations. Formally, consider ℓ independent executions of *SampleNode()*, and for each node $z \in G$ let χ_z^h be the indicator random variable of the event that the h-th execution returns z. If in Equation 7 we replace each random variable χ_z by $\frac{1}{\ell}\sum_{h=1}^{\ell} \chi_z^h$, we obtain the following new estimator for $P(v)$:

$$p_m^\ell(v) = c_m + \frac{1}{\ell}\sum_{h=1}^{\ell}\Big(\sum_{i=1}^{m} \chi_{u_i}^h \cdot c_m(u_i) + \sum_{u:(u,w)\in F(G_m)} \chi_u^h \cdot c_m(u)\Big) \tag{8}$$

Consider now the random variable:

$$\frac{\ell}{c}(p_m^\ell(v) - c_m) = \sum_{h=1}^{\ell}\Big(\sum_{i=1}^{m} \chi_{u_i}^h \cdot \frac{c_m(u_i)}{c} + \sum_{u:(u,w)\in F(G_m)} \chi_u^h \cdot \frac{c_m(u)}{c}\Big) \tag{9}$$

where c is the maximum among all inner and frontier coefficients of $p_m(v)$. Clearly, $p_m^\ell(v)$ is the sum of the nonnegative constant c_m and of $\frac{c}{\ell}$ times $\frac{\ell}{c}(p_m^\ell(v) - c_m)$, and thus deviates by more than a factor $1 \pm \epsilon$ from $P(v)$ only if $\frac{\ell}{c}(p_m^\ell(v) - c_m)$ deviates by more than a factor $(1 \pm \epsilon)$ from its own expectation. Since $\frac{\ell}{c}(p_m^\ell(v) - c_m)$ is by construction a sum of non-positively correlated indicator random variables with weights in $[0, 1]$, the probability of the latter event can be bounded by applying the probability bounds of Appendix A.1, which yield for any ϵ in $[0, 1]$:

$$Pr\big[|p_m^\ell(v) - P(v)| > \epsilon P(v)\big] \leq 2e^{-\epsilon^2 \mathbb{E}[\frac{\ell}{c}(p_m^\ell(v) - c_m)]/3} \tag{10}$$

We can put the above bound in a more telling form by lower bounding the expectation $\mathbb{E}[\frac{\ell}{c}(p_m^\ell(v) - c_m)]$ at its exponent. Since such an expectation is in turn not smaller than the expectation of $\sum_{h=1}^{\ell}\sum_{i=1}^{m} \chi_{u_i}^h \frac{c_m(u_i)}{c}$ (see Equation 9), we can use a lower bound on this latter.

Using the fact that $\mathbb{E}[\chi^h_{u_i}] = P(u_i) \geq \frac{1-\alpha}{n}$ for any u_i, by linearity of expectation we have:

$$\mathbb{E}\Big[\sum_{h=1}^{\ell}\sum_{i=1}^{m}\chi^h_{u_i}\frac{c_m(u_i)}{c}\Big] \geq \ell\, m\, \frac{1-\alpha}{n}\cdot \min_{i=1,\ldots,m}\Big\{\frac{c_m(u_i)}{c}\Big\} \tag{11}$$

The last factor at the right-hand side reflects the dependence of the bound on the inbalance between the coefficients. This relationship is crucial, and as we will see, can be conveniently captured in the following definition:

Definition 1. *We say that $p_m(v)$ is* γ-balanced *if each inner coefficient is at least $1/\gamma$ times any other inner or frontier coefficient.*

If $p_m(v)$ is γ-balanced, then $\frac{c_m(u_i)}{c} \geq 1/\gamma$ for all u_i, and the right-hand side of Equation 11 is at least $\frac{1}{\gamma}\ell m\frac{1-\alpha}{n}$. Plugging this value at the exponent of Equation 10, we obtain:

Lemma 7. *If $p_m(v)$ is γ-balanced, then*

$$Pr\big[|p^\ell_m(v) - P(v)| > \epsilon P(v)\big] \leq 2e^{-(\epsilon^2\frac{1}{\gamma}\ell m\frac{1-\alpha}{n})/3} \tag{12}$$

Lemma 7 highlights the roles of ℓ, m and γ in determining the approximation guarantees given by $p^\ell_m(v)$. Intuitively, if γ is close to 1, then $p^\ell_m(v)$ behaves essentially as the sum of $m\ell$ (or more) non-positively correlated indicator random variables; its variance is small, and with relatively small ℓ and m we can achieve the desired approximation guarantees. On the other hand, if γ is large, one must correspondingly increase ℓ and m (and thus the overall number of queries) to keep valid such guarantees. Note that ℓ has the sole effect of appearing at the exponent of the bound, but is otherwise irrelevant to our analysis; therefore, for the sake of simplicity we will analyse the simpler estimator $p_m(v)$, and recur to Lemma 7 when needed.

Building a 1-balanced estimator. We now show that, if the outdegrees of G are known beforehand, with just $m+1$ queries one can build a 1-balanced estimator $p_m(v)$ – which is essentially optimal. In particular, we show an iterative strategy for selecting the nodes $u_1, \ldots, u_m$ and multipliers $\beta^m_0, \ldots, \beta^m_m$ while building a sequence of 1-balanced estimators $p_1(v), \ldots, p_m(v)$.

The key idea is the following. Recall the expression of $p_m(v)$ in Equation 7. By construction, the multiplier β^m_m appears in every frontier coefficient, but in none of the inner coefficients. Therefore by altering β^m_m (and rescaling $\beta^m_0, \ldots, \beta^m_{m-1}$ so that they add up to $1-\beta^m_m$) one can control the relative ratio between inner and frontier coefficients. Exploiting this fact, we can make the frontier coefficient of some node u' match the inner coefficients; u' is then the next node to be queried, and its term is moved to the inner summation while the frontier summation gains new terms associated to the new frontier nodes that point to u'. In this way we can iteratively build $p_m(v)$ from $p_{m-1}(v)$, by rescaling and moving terms from the frontier summation to the inner summation while always keeping the inner coefficients all identical and never smaller than any frontier coefficient.

Let us describe the iterative step of the construction in more detail. Suppose we have built an estimator $p_{m-1}(v)$ that satisfies the following three conditions: (1) all inner coefficients are identical, (2) all inner coefficients are at least as large as any frontier coefficient, and (3) there exists a frontier coefficient that equals the inner coefficients (conditions (1) and (2) state that $p_{m-1}(v)$ is 1-balanced, while condition (3) is needed by our construction). Starting from $p_{m-1}(v)$, we can build a new estimator $p_m(v)$ that still satisfies all three conditions in the following way. Let u' be the frontier node whose frontier coefficient matches the inner coefficients. We let u_m be u', and move its term from the frontier summation to the inner summation of Equation 7, therefore setting $c_m(u_m) = c_{m-1}(u')$. The inner summation has now m terms, and its m inner coefficients are all identical since by hypothesis on the third condition $c_m(u_m)$ matches all other inner coefficients. We then add to the frontier summation the new terms relative to those nodes

that appear on the frontier of the new subgraph G_m, which we learn by invoking *Neighbourhood(·)* on u_m. To this end, we must pick a new set of nonnegative multipliers $\beta_0^m, \ldots, \beta_m^m$; each β_i^m replaces the corresponding β_i^{m-1} for all $i \leq m-1$, while β_m^m weights the new terms appearing on the frontier summation. Let then $\beta_i^m = \beta_i^{m-1}(1-\beta_m^m)$ for all $i \leq m-1$. We are in other words rescaling all inner coefficients by a factor $(1-\beta_m^m)$, and thus keeping condition (1) satisfied. On the other hand, any frontier coefficient $c_m(u)$ is the sum of $c_{m-1}(u)$, rescaled by $(1-\beta_m^m)$, and of a new term proportional to β_m^m; hence if we gradually increase β_m^m starting from 0, some frontier coefficient will eventually match the inner coefficients. At this point, conditions (2) and (3) are also both satisfied. Note that the construction requires computing the frontier coefficients, which depend on the outdegrees of the frontier nodes. We can thus perform the construction if we know the outdegree of each node of the graph in advance. Appendix A.8 gives a formal description and analysis of the construction, proving:

Lemma 8. *If the outdegree of every node in G is known beforehand, then with $m+1$* Neighbourhood(·) *queries one can build a 1-balanced estimator $p_m(v)$.*

Coupled with Lemma 7, Lemma 8 implies that, if the outdegrees of G are known, then with $m+1$ *Neighbourhood(·)* queries one can ensure that $Pr\big[|p_m^\ell(v) - P(v)| > \epsilon P(v)\big] \leq 2e^{-(\epsilon^2 \ell m \frac{1-\alpha}{n})/3}$. The right-hand side can be made arbitrarily smaller than any $\delta > 0$ by choosing $m, \ell = O(n^{\frac{1}{2}}\sqrt{\ln(1/\delta)}\frac{1}{\epsilon})$, i.e. $O(\sqrt{n})$ queries suffice to estimate $P(v)$ if one knows the outdegrees of G in advance. The next section shows that this is in fact a special case of a more general scenario where one has only *approximate* knowledge of the outdegrees.

3.3 Using approximate knowledge of the outdegrees

This section generalizes the construction of the 1-balanced estimator of Subsection 3.2 in the case the outdegrees of the graph are not known in advance. As we will see, the concentration bounds of the resulting estimator actually depend on prior knowledge of the *relative ratios* between those outdegrees, which will lead us to prove Theorem 2.

Suppose then the outdegrees of G are initially unknown. In such a case, we cannot carry out the construction of the sequences $u_1, \ldots, u_m$ and $\beta_0^m, \ldots, \beta_m^m$ described in Subsection 3.2, since at each step it requires computing the frontier coefficients $c_m(u)$, which in turn requires knowledge of the outdegrees of the frontier nodes (which are unknown). Suppose however we dispose of an estimate $\widehat{out}(u)$ of the outdegree $out(u)$ of each node u in G, before the construction begins. We then proceed through the construction, using the surrogate inner coefficients $\hat{c}_m(u_i) = \sum_{j=j(u_i)}^{i-1} \beta_j^m \frac{c_{G_j}(u_i)}{\widehat{out}(u_i)}$ and frontier coefficients $\hat{c}_m(u) = \sum_{j=j(u)}^{m} \beta_j^m \frac{c_{G_j}(u)}{\widehat{out}(u)}$ in place of the actual ones. Clearly, the selected $u_1, \ldots, u_m$ and $\beta_0^m, \ldots, \beta_m^m$ now produce inner coefficients and frontier coefficients that are no longer balanced in the sense of Lemma 8; however, by the same argument of Lemma 8, the *surrogate* coefficients can be balanced, i.e. all $\hat{c}_m(u_i)$ can be made identical and at least as large as any $\hat{c}_m(u)$. Then, since $c_m(u_i) = \frac{\widehat{out}(u_i)}{out(u_i)}\hat{c}_m(u_i)$ and $c_m(u) = \frac{\widehat{out}(u)}{out(u)}\hat{c}_m(u)$, the imbalance among the actual coefficients is bounded by how much the estimates "distort" the ratios between the actual outdegrees.

In fact, it is not hard to prove that the construction can be carried out even if one only disposes of estimates of those ratios, instead of estimates of the outdegrees themselves. This intuitive argument is formalized in the following generalized version of Lemma 8, which follows imediately from the proof of Lemma 8 and the definitions of $\hat{c}_m(u_i)$ and $\hat{c}_m(u)$:

Lemma 9. *If for any pair of nodes $u, u' \in G$ with positive outdegrees one has an estimate of $out(u)/out(u')$ which is at most γ times the actual value, then with $m+1$* Neighbourhood(·) *queries one can build a γ-balanced estimator $p_m(v)$.*

Lemma 9 gives a key fact: an approximation factor γ in the knowledge of the relative ratios of the outdegrees allows one to build a γ-balanced estimator. By Lemma 7 this also implies that, by choosing $m, \ell = O\big(n^{\frac{1}{2}}\gamma^{\frac{1}{2}}\sqrt{\ln(1/\delta)}\frac{1}{\epsilon}\big)$, one can make arbitrarily smaller than any $\delta > 0$ the

probability of erring by more than a multiplicative factor $(1 \pm \epsilon)$ in the estimate of $P(v)$, thus proving Theorem 2.

Before continuing, let us make a crucial observation on how to actually take a sample of $p_m(v)$ (the same applies to $p_m^\ell(v)$). Once we have chosen $u_1, \ldots, u_m$ and $\beta_0^m, \ldots, \beta_m^m$, we can readily compute c_m and the inner coefficients; but we cannot compute the frontier coefficients, since their denominators depend on the unknown outdegrees of the nodes on the frontier of G_m, which could be $\Theta(n)$. This means we cannot in general compute the explicit expression of $p_m(v)$ with less than $\Theta(n)$ queries. Observe however that a single realization of $p_m(v)$ simply equals $c_m + c_m(z)$, where z is the node returned by *SampleNode()*; all other terms are set to zero by their corresponding indicator random variables. Therefore, after invoking *SampleNode()*, we just need to retrieve $out(z)$ (by invoking *Neighbourhood(z)*) to compute $c_m(z)$ and the realization of $p_m(v)$, without knowing its explicit expression.

3.4 Approximating the outdegrees of the graph

The last ingredient in the construction of our algorithm is a randomized scheme for approximating the positive outdegrees of G. We show how, using roughly k queries, one can (probabilistically) obtain an approximation of such outdegrees accurate enough to ensure γ is roughly $\frac{n \log n}{k}$, which will then lead us to prove Theorem 1.

We start from a simple observation: for any u in G, a node picked uniformly at random in G has probability $\frac{out(u)}{n}$ of being a child of u. Consider then sampling k nodes independently and uniformly at random from G, and for each u in G let ψ_u^j be the indicator random variable of the event that the j-th node sampled is a child of u. Clearly $\mathbb{E}[\frac{n}{k} \sum_{j=1}^k \psi_u^j]$ equals $out(u)$, and therefore $\frac{n}{k} \sum_{j=1}^k \psi_u^j$ is an unbiased estimator of $out(u)$. Unfortunately, such an estimator has two limitations. First, while by standard concentration bounds any outdegree of the order of $\frac{n}{k}$ or larger (i.e. for which $\mathbb{E}[\sum_{j=1}^k \psi_u^j] = \Omega(1)$) will likely be well estimated, smaller outdegrees will likely receive an estimate equal to 0, making γ ill-defined. Second, even if we make the probability of receiving a poor estimate small for any given node in G, unless such a probability is polynomially small in n, the probability that *some* node will receive a poor estimate will potentially be large. We solve both issues by adding a constant quantity $\frac{n}{k} b \log(n)$ to the estimator, therefore setting $\widehat{out}(u) = \frac{n}{k}(b \log(n) + \sum_{j=1}^k \psi_u^j)$. Now, on the one hand, any positive outdegree not exceeding $\frac{n}{k} b \log(n)$ is never underestimated, and can be overestimated by a multiplicative factor at most $2\frac{n}{k} b \log(n)$ unless $\sum_{j=1}^k \psi_u^j > b \log(n)$, which by standard concentration bounds happens with probability $O(n^{-b})$. On the other hand, for any outdegree larger than $\frac{n}{k} b \log(n)$ we have $\mathbb{E}[\sum_{j=1}^k \psi_u^j] > b \log(n)$ and thus, by standard concentration bounds, $\widehat{out}(u)$ deviates significantly from $out(u)$ with probability only $O(n^{-b})$. This intuitive argument is formalized in the following lemma, proved in Appendix A.9:

Lemma 10. *With k* RandomNode() *and k* Neighbourhood($\cdot$) *queries one can obtain estimates $\{\widehat{out}(u)\}_{u \in G}$ such that, for any $b > 0$, we have $\gamma \leq 4b\frac{n \log(n)}{k}$ with probability $1 - 2n^{-\frac{b}{8 \ln(2)} + 1}$.*

We can now conclude the proof of Theorem 1. By Lemma 10, we can make arbitrarily smaller than δ the probability that $\gamma > 4b\frac{n \log(n)}{k}$ by choosing a large enough $b \in O(1 + \ln(1/\delta)/\ln(n)) = O(\ln(n/\delta)/\ln(n))$. Then, if $\gamma \leq 4b\frac{n \log(n)}{k}$, by Lemma 9 and Lemma 7 we obtain:

$$Pr\big[|p_m^\ell(v) - P(v)| > \epsilon P(v)\big] \leq 2e^{-(\epsilon^2 \frac{k\ell m}{4bn \log(n)} \frac{1-\alpha}{n})/3} \tag{13}$$

To make the right-hand side arbitrarily smaller than δ, it suffices to choose k, ℓ, m large enough and such that $k\ell m \in O\big(\frac{bn^2 \log(n)}{\epsilon^2} \ln(1/\delta)\big)$. We can then set $k = \ell = m$ to minimize the overall number of queries $O(k + \ell + m)$; together with the fact that $b \in O(\ln(n/\delta)/\ln(n))$, this gives

$$k = \ell = m \in O\Big(n^{\frac{2}{3}} \sqrt[3]{\log(n/\delta)}\, \ln(1/\delta)^{\frac{1}{3}} \epsilon^{-\frac{2}{3}}\Big) = \tilde{O}\big(n^{\frac{2}{3}}\big) \tag{14}$$

References

[1] R. Andersen, C. Borgs, J. Chayes, J. Hopcroft, V. Mirrokni, and S.-H. Teng. Local computation of PageRank contributions. *Internet Mathematics*, 5(1–2):23–45, 2008.

[2] A. Auger and B. Doerr (eds.). *Theory of Randomized Search Heuristics: Foundations and Recent Developments*, vol. 1 of *Series on Theoretical Computer Science*. World Scientific Publishing Co., Inc., 2011.

[3] 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*, 45(2):890–904, 2007.

[4] B. Bahmani, R. Kumar, M. Mahdian, and E. Upfal. PageRank on an evolving graph. In *Proc. of ACM KDD*, pages 24–32. 2012.

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

[6] Z. Bar-Yossef and L.-T. Mashiach. Local approximation of PageRank and reverse PageRank. In *Proc. of ACM CIKM*, pages 279–288. 2008.

[7] Z. Bar-Yossef and L.-T. Mashiach. Local approximation of PageRank and reverse PageRank. In *Proc. of ACM SIGIR*, pages 865–866. 2008.

[8] C. Borgs, M. Brautbar, J. T. Chayes, and S.-H. Teng. Multiscale matrix sampling and sublinear-time PageRank computation. *Internet Mathematics*, 10(1-2):20–48, 2014.

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

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

[11] M. Bressan, E. Peserico, and L. Pretto. Approximating PageRank locally with sublinear query complexity. CoRR abs/1404.1864, April 2014. `http://arxiv.org/abs/1404.1864`.

[12] M. Bressan, E. Peserico, and L. Pretto. The power of local information in PageRank. CoRR abs/1604.00202, April 2016. `http://arxiv.org/abs/1604.00202`.

[13] M. Bressan and L. Pretto. Local computation of PageRank: the ranking side. In *Proc. of ACM CIKM*, pages 631–640. 2011.

[14] S. Brin and L. Page. The anatomy of a large scale hypertextual Web search engine. In *Proc. of WWW*. 1998.

[15] Y.-Y. Chen, Q. Gan, and T. Suel. Local methods for estimating PageRank values. In *Proc. of ACM CIKM*, pages 381–389. 2004.

[16] F. Chierichetti, A. Dasgupta, R. Kumar, S. Lattanzi, and T. Sarlós. On Sampling Nodes in a Network. In *Proc. of WWW*, pages 471–481. 2016.

[17] A. Dasgupta, R. Kumar, and T. Sarlós. On Estimating the Average Degree. In *Proc. of WWW*, pages 795–806. 2014.

[18] D. Fogaras, B. Rácz, K. Csalogány, and T. Sarlós. Towards scaling fully personalized PageRank: Algorithms, lower bounds, and experiments. *Internet Mathematics*, 2(3):333–358, 2005.

[19] Y. Fujiwara, M. Nakatsuji, T. Yamamuro, H. Shiokawa, and M. Onizuka. Efficient personalized PageRank with accuracy assurance. In *Proc. of ACM KDD*, pages 15–23. 2012.

[20] D. Gleich and M. Polito. Approximating personalized PageRank with minimal use of web graph data. *Internet Mathematics*, 3(3):257–294, 2007.

[21] Z. Gyöngyi, H. Garcia-Molina, and J. Pedersen. Combating web spam with TrustRank. In *Proc. of VLDB*, pages 576–587. 2004.

[22] O. Goldreich, S. Goldwasser, and D. Ron. Property testing and its connection to learning and approximation. *Journal of the ACM*, 45(4):653–750, 1998.

[23] O. Goldreich and D. Ron. Property testing in bounded degree graphs. *Algorithmica*, 32(2):302–343, 2002.

[24] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina. The EigenTrust algorithm for reputation management in P2P networks. In *Proc. of WWW*, pages 508–516. 2003.

[25] R. Kumar, P. Raghavan, S. Rajagopalan, D. Sivakumar, A. Tomkins, and E. Upfal. Stochastic models for the web graph. In *Proc. of IEEE FOCS*, pages 57–65. 2000.

[26] C. E. Lee, A. Ozdaglar, and D. Shah. Computing the stationary distribution, locally. In *Proc. of NIPS*, pages 1376–1384. 2013.

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

[28] P. Lofgren, S. Banerjee, and A. Goel. Personalized PageRank estimation and search: A bidirectional approach. In *Proc. of ACM WSDM*, pages 163–172. 2016.

[29] P. Lofgren, S. Banerjee, A. Goel, and C. Seshadhri. FAST-PPR: Scaling Personalized PageRank estimation for large graphs. CoRR abs/1404.3181, April 2014. `http://arxiv.org/abs/1404.3181`

[30] P. Lofgren, S. Banerjee, A. Goel, and C. Seshadhri. FAST-PPR: Scaling Personalized PageRank estimation for large graphs. In *Proc. of ACM KDD*, pages 1436–1445. 2014.

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

[32] L. Page, S. Brin, R. Motwani, and T. Winograd. The PageRank citation ranking: Bringing order to the web. Technical report, Stanford University. 1998.

[33] A. Panconesi and A. Srinivasan. Randomized distributed edge coloring via an extension of the Chernoff-Hoeffding bounds. *SIAM Journal on Computing*, 26(2):350–368, 1997.

[34] Z. E. Schnabel. The estimation of total fish population of a lake. *The American Mathematical Monthly*, 45(6):348–352, 1938.

[35] D. A. Spielman and S.-H. Teng. A local clustering algorithm for massive graphs and its application to nearly linear time graph partitioning. *SIAM Journal on Computing*, 42(1):1–26, 2013.

[36] P. Tarau, R. Mihalcea, and E. Figa. Semantic document engineering with WordNet and PageRank. In *Proc. of ACM SAC*, pages 782–786. 2005.

[37] Z. A. Zhu, S. Lattanzi, and V. S. Mirrokni. A local algorithm for finding well-connected clusters. In *Proc. of ICML (3rd cycle)*, pages 396–404. 2013.

A Appendix

A.1 Probability bounds

This appendix provides Chernoff-type probability bounds that are repeatedly used in our analysis; these bounds can be found in e.g. [2], and can be derived from [33].

Let $X_1, \ldots, X_n$ be binary random variables. We say that $X_1, \ldots, X_n$ are non-positively correlated if for all $I \subseteq \{1, \ldots, n\}$ we have:

$$Pr[\forall i \in I : X_i = 0] \leq \prod_{i \in I} Pr[X_i = 0] \tag{15}$$

$$Pr[\forall i \in I : X_i = 1] \leq \prod_{i \in I} Pr[X_i = 1] \tag{16}$$

The following lemma holds:

Lemma 11. *Let $X_1, \ldots, X_n$ be independent or, more generally, non-positively correlated binary random variables. Let $a_1, \ldots, a_n \in [0, 1]$ and $X = \sum_{i=1}^{n} a_i X_i$. Then, for any $\epsilon > 0$, we have:*

$$Pr[X < (1 - \epsilon)\mathbb{E}[X]] < e^{-\frac{\epsilon^2}{2}\mathbb{E}[X]} \tag{17}$$

$$Pr[X > (1 + \epsilon)\mathbb{E}[X]] < e^{-\frac{\epsilon^2}{2+\epsilon}\mathbb{E}[X]} \tag{18}$$

Note that Lemma 11 applies if $X_1, \ldots, X_n$ are indicator variables of mutually disjoint events, or if they can be partitioned into independent families $\{X_1, \ldots, X_{i_1}\}$, $\{X_{i_1+1}, \ldots, X_{i_2}\}$, ... of such variables.

A.2 A *RandomNode()*-based routine for approximating n

We present a routine, based on repeated invocations of *RandomNode()*, for estimating the number n of nodes in a graph. We show that, for any given $\epsilon, \delta > 0$, with probability $(1 - \delta)$ our routine returns an estimate of n accurate within a factor $(1 \pm \epsilon)$ invoking *RandomNode()* at most $O\big(n^{\frac{1}{2}}\sqrt{\ln(1/\delta)}\frac{1}{\epsilon}\big)$ times. Although estimating the cardinality of a set via random sampling is a classic problem in statistics (see e.g. [34]), the present result has the advantages of being self-contained and of providing bounds in the form needed by the bounds of Theorem 2.

Algorithm CardinalityEstimator(ϵ, δ)

1: $s \leftarrow 0$ ▷ number of samples taken so far
2: $d \leftarrow 0$ ▷ number of *distinct* samples taken so far
3: $w \leftarrow 0$ ▷ number of samples taken so far, each weighted by d when taken
4: $k_{\epsilon,\delta} \leftarrow \frac{4}{\epsilon^2} \ln\left(\frac{3}{\delta}\right)$ ▷ halting threshold on the number of repeated samples
5: **repeat**
6: $\quad w \leftarrow w + d$
7: $\quad s \leftarrow s + 1$
8: $\quad sample[s] \leftarrow$*RandomNode()*
9: $\quad$ **if** $sample[s] \notin \bigcup_{i=1}^{s-1} sample[i]$ **then** $d \leftarrow d + 1$
10: **until** $(s - d) \geq k_{\epsilon,\delta}$
11: **return** $\frac{w}{s-d}$

Theorem 12. CardinalityEstimator *with probability greater than $(1 - \delta)$ outputs an estimate $\hat{n}$ such that $(1 - \epsilon)n \leq \hat{n} \leq (1 + \epsilon)n$ invoking* RandomNode() *at most $O\big(n^{\frac{1}{2}}\sqrt{\ln(1/\delta)}\frac{1}{\epsilon}\big)$ times.*

Proof. Suppose CardinalityEstimator has so far taken exactly $i - 1$ samples, and denote by $d(i)$ and $w(i)$ the values of d and w just before the i^{th} sample is taken (i.e. just before line 7 is executed

for the i^{th} time). The probability that the next invocation of *RandomNode()* yields a repeat is then $\frac{d(i)}{n}$. Therefore, although clearly $d(i)$ is determined by the outcomes of previous samples, once its value is known we can associate to the event that the next invocation of *RandomNode()* yields a repeat an indicator random variable $\chi_i^{d(i)}$ with expectation $\frac{d(i)}{n}$; and, crucially, this random variable is independent from $\chi_1^{d(1)}, \ldots, \chi_{i-1}^{d(i-1)}$. Therefore, whenever line 9 is executed, the number $s-d$ of repeats witnessed so far by the algorithm is the realization of the sum of s independent indicator random variables $\chi_1^{d(1)}, \ldots, \chi_s^{d(s)}$ with total expectation

$$\mathbb{E}\Big[\sum_{i=1}^{s} \chi_i^{d(i)}\Big] = \sum_{i=1}^{s} \frac{d(i)}{n} = \frac{w(s)}{n} \tag{19}$$

At line 10, the algorithm returns as an estimate of n the value $\hat{n}$ that would make the formula for $\mathbb{E}\big[\sum_{i=1}^{s} \chi_i^{d(i)}\big]$ given by Equation 19 match the actual value $s-d$ of the realization of $\sum_{i=1}^{s} \chi_i^{d(i)}$, i.e. $\hat{n} = \frac{w(s)}{s-d}$; for this estimate to deviate excessively from n we must then have $\mathbb{E}\big[\sum_{i=1}^{s} \chi_i^{d(i)}\big]$ far enough from the actual realization $s - d \geq k_{\epsilon,\delta}$ of $\sum_{i=1}^{s} \chi_i^{d(i)}$. In a nutshell, the algorithm stops acquiring samples when the sum of the realizations of the (independent) $\chi_i^{d(i)}$ is sufficiently large to make a significant deviation from its expectation unlikely.

Let us begin by bounding the probability that the estimate exceeds $(1+\epsilon)n$. For this event to take place, the algorithm must terminate with $\frac{w(s)}{s-d} = (1+\bar{\epsilon})n$ for some $\bar{\epsilon} > \epsilon$. Thus $\mathbb{E}\big[\sum_{i=1}^{s} \chi_i^{d(i)}\big] = \frac{w(s)}{n} = (1+\bar{\epsilon})(s-d) \geq (1+\bar{\epsilon})k_{\epsilon,\delta}$, and $\sum_{i=1}^{s} \chi_i^{d(i)}$ has taken value $s - d = (1 - \frac{\bar{\epsilon}}{1+\bar{\epsilon}})\mathbb{E}\big[\sum_{i=1}^{s} \chi_i^{d(i)}\big]$. By the bounds of Appendix A.1 (still valid when $<$ is replaced by $\leq$) the probability of this event is then at most

$$Pr\Big[\sum_{i=1}^{s} \chi_i^{d(i)} \leq \big(1 - \frac{\bar{\epsilon}}{1+\bar{\epsilon}}\big)\mathbb{E}\Big[\sum_{i=1}^{s} \chi_i^{d(i)}\Big]\Big] \leq e^{-\frac{(1+\bar{\epsilon})k_{\epsilon,\delta}(\frac{\bar{\epsilon}}{1+\bar{\epsilon}})^2}{2}} = \Big(\frac{\delta}{3}\Big)^{\frac{2\bar{\epsilon}^2}{\epsilon^2(1+\bar{\epsilon})}} \tag{20}$$

and the last term is smaller than $\frac{\delta}{3}$ since $\frac{2\bar{\epsilon}^2}{\epsilon^2(1+\bar{\epsilon})} \geq \frac{\bar{\epsilon}^2}{\epsilon^2} > 1$ when $\bar{\epsilon} \leq 1$ and $\frac{2\bar{\epsilon}^2}{\epsilon^2(1+\bar{\epsilon})} > \frac{\bar{\epsilon}}{\epsilon^2} > 1$ when $\bar{\epsilon} > 1$.

In a similar fashion we can bound the probability that the estimate falls below $(1-\epsilon)n$. In this case $\frac{w(s)}{s-d} = (1-\bar{\epsilon})n$ for some $1 > \bar{\epsilon} > \epsilon$, thus $\mathbb{E}\big[\sum_{i=1}^{s} \chi_i^{d(i)}\big] = \frac{w(s)}{n} = (1-\bar{\epsilon})(s-d) \geq (1-\bar{\epsilon})k_{\epsilon,\delta}$ and $\sum_{i=1}^{s} \chi_i^{d(i)}$ has taken value $s-d = (1+\frac{\bar{\epsilon}}{1-\bar{\epsilon}})\mathbb{E}\big[\sum_{i=1}^{s} \chi_i^{d(i)}\big]$. Again by the bounds of Appendix A.1, the probability of this event is no larger than

$$Pr\Big[\sum_{i=1}^{s} \chi_i^{d(i)} \geq \big(1 + \frac{\bar{\epsilon}}{1-\bar{\epsilon}}\big)\mathbb{E}\Big[\sum_{i=1}^{s} \chi_i^{d(i)}\Big]\Big] \leq e^{-\frac{(1-\bar{\epsilon})k_{\epsilon,\delta}(\frac{\bar{\epsilon}}{1-\bar{\epsilon}})^2}{2+\frac{\bar{\epsilon}}{1-\bar{\epsilon}}}} = \Big(\frac{\delta}{3}\Big)^{\frac{4\bar{\epsilon}^2}{\epsilon^2(2-\bar{\epsilon})}} \tag{21}$$

and the last term is smaller than $\frac{\delta}{3}$ since $\frac{4\bar{\epsilon}^2}{\epsilon^2(2-\bar{\epsilon})} > \frac{\bar{\epsilon}^2}{\epsilon^2} > 1$.

Let us now prove less than $\frac{\delta}{3}$ the probability that the algorithm invokes *RandomNode()* more than $2\lceil\sqrt{k_{\epsilon,\delta}\, n}\,\rceil + k_{\epsilon,\delta} = O\big(n^{\frac{1}{2}}\sqrt{\ln(1/\delta)}\frac{1}{\epsilon}\big)$ times before obtaining $k_{\epsilon,\delta}$ repeats and thus terminating. Clearly, this event takes place if and only if *RandomNode()* yields $2\lceil\sqrt{k_{\epsilon,\delta}\, n}\,\rceil + 1$ distinct samples before yielding $k_{\epsilon,\delta}$ repeats, and thus, denoting by $r(d)$ the random variable giving the total number of repeats yielded by *RandomNode()* before the $(d+1)^{th}$ distinct sample is obtained, if and only if $r(2\lceil\sqrt{k_{\epsilon,\delta}\, n}\,\rceil) < k_{\epsilon,\delta}$. Denote by ρ_i the indicator variable of the event of obtaining at least one repeat between the i^{th} and $(i+1)^{th}$ distinct samples; note that obviously $r(d) \geq \sum_{i=1}^{d} \rho_i$, that all ρ_i are independent, and that $\mathbb{E}[\rho_i] = \frac{i}{n}$ and thus $\mathbb{E}\big[\sum_{i=1}^{2\lceil\sqrt{k_{\epsilon,\delta} n}\rceil} \rho_i\big] =$

$\sum_{i=1}^{2\lceil\sqrt{k_{\epsilon,\delta}n}\rceil} \frac{i}{n} > 2k_{\epsilon,\delta}$. Then, once again by the bounds of Appendix A.1,

$$Pr\Big[r\big(2\lceil\sqrt{k_{\epsilon,\delta}\,n}\,\rceil\big) < k_{\epsilon,\delta}\Big] < e^{-\frac{(1/2)^2}{2}\cdot 2k_{\epsilon,\delta}} = e^{-\frac{1}{\epsilon^2}\ln\frac{3}{\delta}} < \frac{\delta}{3} \tag{22}$$

Since the probabilities that the estimate returned by CardinalityEstimator falls below $(1-\epsilon)n$ or exceeds $(1+\epsilon)n$ are also each less than $\frac{\delta}{3}$, by a simple union bound we obtain the thesis. □

A.3 A formal definition of PageRank conductance

Given two nodes u and v in a graph G, a *path* π from u to v is a sequence of arcs of G such that either $\pi = \emptyset$ and $u = v$, or π is the concatenation of a path π' from u to some node u' such that $(u', v) \in G$ with the arc (u', v). We denote by $|\pi|$ the length of π, i.e. the number of arcs in it. Also, we denote by $out(w)$ the outdegree of node w. Then:

Definition 2. *The PageRank conductance of a path π is:*

$$\mho^{\pi} = \prod_{(w,w')\in\pi} \frac{\alpha}{out(w)} = \alpha^{|\pi|} \prod_{(w,w')\in\pi} \frac{1}{out(w)} \tag{23}$$

Given a path π and a subgraph G' of G, we say that π is in G' if the arcs of π are all arcs of G'. Let then $\Pi_{G'}(u, v)$ denote the set of all the paths from u to v that are in G'. The *PageRank conductance* of G' from u to v is the sum of the PageRank conductances of all paths in $\Pi_{G'}(u, v)$:

Definition 3. *Given two nodes u and v in G and a subgraph G' of G, the PageRank conductance $\mho_{G'}(u, v)$ of G' from u to v is:*

$$\mho_{G'}(u, v) = \sum_{\pi\in\Pi_{G'}(u,v)} \mho^{\pi} \tag{24}$$

A useful interpretation of $\mho_{G'}(u, v)$ can be derived by breaking the summation in Equation 24 over all possible path lengths:

$$\mho_{G'}(u, v) = \sum_{k\geq 0} \sum_{\substack{\pi\in\Pi_{G'}(u,v)\\ |\pi|=k}} \mho^{\pi} \tag{25}$$

The inner summation is the probability that the PageRank random walk reaches v from u in k steps without ever leaving G' or taking a random jump. Therefore, $\mho_{G'}(u, v)$ is the expected number of times a walk starting in u goes through v before leaving G' or taking a random jump.

Note that $\mho_{G'}(u, v)$ is always finite, since the probability that the random walk follows some path of length k (and thus the inner summation in Equation 25) is bounded by α^k. Note also that in any G' there always exists a path from a node to itself, the empty path of length 0, whose PageRank conductance is exactly 1. Finally, note that the PageRank conductance between two nodes is monotonically non-decreasing with the subgraph G', that is, if $G' \subseteq G'' \subseteq G$ then $\mho_{G'}(u, v) \leq \mho_{G''}(u, v)$, since $\Pi_{G'}(u, v) \subseteq \Pi_{G''}(u, v)$.

A.4 Proof of Lemma 5

Proof. Recall the PageRank random walk on the nodes of G defined in Subsection 1.1, and consider it at the stationary state so that at any given instant the walk is in u with probability $P(u)$. Suppose the walk is currently in v, and thus by hypothesis in $\bar{G}$. Clearly, since the last random jump performed in the past, the walk has either been only in nodes of $\bar{G}$ or has also been in some node of $G \setminus \bar{G}$. We compute $P(v)$ by analysing these two cases separately. We use the notions of path, path in a subgraph, and PageRank conductance of a path, as well as the related notation – see Appendix A.3.

Let us start by computing the probability of the event $\mathcal{E}_v^{\in\bar{G}}$ that the walk is in v having always been in nodes of $\bar{G}$ since the last random jump. This event takes place if and only if, for some $k \geq 0$, the random walk performed a random jump k steps ago, this jump brought the walk to some node $w \in \bar{G}$, and the walk then followed a path π in $\bar{G}$ of length k from w to v. The probability of performing a random jump is $(1-\alpha)$ at any instant, the probability that the jump leads to any given node is $\frac{1}{n}$, and the probability of following a path π is its PageRank conductance $\mho^\pi$. Therefore $P(\mathcal{E}_v^{\in\bar{G}})$ is the sum of the product of these three terms over all $k \geq 0$, all $w \in \bar{G}$, and all paths of length k from w to v in $\bar{G}$:

$$P(\mathcal{E}_v^{\in\bar{G}}) \;=\; \sum_{k\geq 0}(1-\alpha)\sum_{w\in\bar{G}}\frac{1}{n}\sum_{\substack{\pi\in\Pi_{\bar{G}}(w,v)\\|\pi|=k}}\mho^\pi \;=\; \sum_{w\in\bar{G}}\frac{1-\alpha}{n}\,\mho_{\bar{G}}(w,v) \tag{26}$$

where we used the fact that $\sum_{\pi\in\Pi_{\bar{G}}(w,v)}\mho^\pi = \mho_{\bar{G}}(w,v)$.

Let us now compute the probability of the event $\mathcal{E}_v^{\notin\bar{G}}$ that the walk is in v having been in some node of $G\setminus\bar{G}$ since the last random jump. This event takes place if and only if the walk enters $\bar{G}$ from $G\setminus\bar{G}$ through a frontier arc, and then follows a path entirely in $\bar{G}$ to v. Thus, for some $k \geq 0$ the walk was $k+1$ steps ago in a node u having some outgoing arcs in $F(\bar{G})$, then followed one of these arcs (u,w) to a node $w \in \bar{G}$, and finally followed a path π in $\bar{G}$ of length k from w to v. The probability of being in u is $P(u)$ at any instant, the probability of following a frontier arc (u,w) is $\frac{\alpha}{out(u)}$ for any given w, and the probability of following any given path π is $\mho^\pi$. Similarly to above, then, we obtain $P(\mathcal{E}_v^{\notin\bar{G}})$ by summing probabilities over all relevant variables:

$$P(\mathcal{E}_v^{\notin\bar{G}}) = \sum_{k\geq 0}\sum_{(u,w)\in F(\bar{G})}P(u)\,\frac{\alpha}{out(u)}\sum_{\substack{\pi\in\Pi_{\bar{G}}(w,v)\\|\pi|=k}}\mho^\pi \qquad = \sum_{(u,w)\in F(\bar{G})}P(u)\frac{\alpha}{out(u)}\,\mho_{\bar{G}}(w,v) \tag{27}$$

Since $P(v) = P(\mathcal{E}_v^{\in\bar{G}}) + P(\mathcal{E}_v^{\notin\bar{G}})$, summing Equations 26 and 27 proves the thesis. □

A.5 Formal definition of the routine *SampleNode()*

Algorithm SampleNode()

```
current_node ← RandomNode()
loop
    with probability (1 − α) return current_node
    current_node ← RandomChild(current_node)
    if current_node = ∅ then current_node ← RandomNode()
end loop
```

A.6 Proof of Lemma 4

Consider the random walk used in Subsection 1.1 to define PageRank in terms of an abstract surfer. We can imagine the surfer selects at any node, whether dangling or not, with probability $(1-\alpha)$ a random jump leading to a node chosen uniformly at random, and with probability α either a *virtual arc* leading to a node chosen uniformly at random if the current node is dangling, or an actual arc leading to a child (chosen uniformly at random) of the current node otherwise. The stationary probability of the latest jump being τ steps in the past (i.e. of having followed exactly τ arcs, whether virtual or actual, after it) is $(1-\alpha)\alpha^\tau$. Since *SampleNode()* follows, starting from a node chosen uniformly at random, exactly τ arcs (virtual or actual) with probability $(1-\alpha)\alpha^\tau$, it returns node v with probability equal to $P(v)$.

A.7 A bound on the query complexity of *SampleNode()*

Lemma 13. *The probability that* $\frac{2\ell}{(1-\epsilon)(1-\alpha)}$ RandomNode() *and* RandomChild($\cdot$) *queries are not sufficient to complete* ℓ *calls to* SampleNode() *is less than* $e^{-\ell\epsilon^2/2(1-\epsilon)}$.

Proof. If a sequence of *SampleNode()* calls performs q queries, then line 3 is executed at least $q/2$ times, each time making *SampleNode()* terminate independently with probability $(1-\alpha)$. The expected number of *SampleNode()* calls one can complete with q queries is thus at least $q(1-\alpha)/2$, and by the bounds of Appendix A.1, the probability that the number of completed calls does not reach $(1-\epsilon)q(1-\alpha)/2$ is less than $e^{-q(1-\alpha)\epsilon^2/4}$. Setting $q = \frac{2\ell}{(1-\epsilon)(1-\alpha)}$ gives the thesis. □

A.8 Proof of Lemma 8

We prove a slightly stronger result anticipated in the sketch and needed by the proof: if the outdegree of every node in G is known beforehand, then with $m+1$ *Neighbourhood($\cdot$)* queries one can choose $u_1, \ldots, u_m$ and $\beta_0^m, \beta_1^m, \ldots, \beta_m^m$ so that $p_m(v)$ is 1-balanced and that there exists a frontier node whose coefficient matches the inner coefficients. More formally, we prove one can make the coefficients of $p_m(v)$ simultaneously satisfy the following three conditions:

$$\forall i, i' : 1 \le i, i' \le m : \qquad c_m(u_i) = c_m(u_{i'}) \tag{28}$$

$$\forall u : (u, w) \in F(G_m) : \qquad c_m(u) \le c_m(u_m) \tag{29}$$

$$F(G_m) \neq \emptyset \implies \exists u : (u, w) \in F(G_m) : \qquad c_m(u) = c_m(u_m) \tag{30}$$

Let us start with the case $m = 1$. Equation 28 is trivially satisfied, regardless of the choice of u_1, and we can focus on the second and third conditions. Suppose then $F(G_0)$ is nonempty, for otherwise we can stop the construction and compute $P(v)$, and let u_1 be the parent of v with the smallest outdegree (breaking ties arbitrarily). Recalling the definitions of $c_m(u_i)$ and $c_m(u)$, we can then write:

$$c_1(u_1) = (1 - \beta_1^1)\frac{c_{G_0}(u_1)}{out(u_1)} \tag{31}$$

$$c_1(u) = \sum_{j=j(u)}^{0} (1 - \beta_1^1)\frac{c_{G_0}(u)}{out(u)} \; + \; \beta_1^1 \frac{c_{G_1}(u)}{out(u)} \qquad \forall u : (u, w) \in F(G_1) \tag{32}$$

where we used the fact that $\beta_0^1 = 1 - \beta_1^1$ and split $c_1(u)$ into the terms relative to $j = 0$ and $j = 1$. Since all arcs in $F(G_0)$ point to the same node v, then for all $u : (u, w) \in F(G_0)$ we have $c_{G_0}(u_1) = c_{G_0}(u)$, and by choice of u_1 we have $\frac{c_{G_0}(u)}{out(u)} \le \frac{c_{G_0}(u_1)}{out(u_1)}$. Therefore $c_1(u) \le c_1(u_1)$ when $\beta_1^1 = 0$, while $0 = c_1(u_1) < c_1(u)$ when $\beta_1^1 = 1$, for all $u : (u, w) \in F(G_1)$. By the continuity of $c_1(u_1)$ and $c_1(u)$ as functions of β_1^1, we conclude that there exists a value of β_1^1 in $[0, 1)$ which makes the coefficients simultaneously satisfy also equations 29 and 30.

Suppose now we have chosen $u_1, \ldots, u_{m-1}$ and $\beta_0^{m-1}, \beta_1^{m-1}, \ldots, \beta_{m-1}^{m-1}$ that satisfy equations 28, 29 and 30. Again, suppose $F(G_{m-1})$ is nonempty, for otherwise we can stop the construction and compute $P(v)$. Consider the sets $u_1, \ldots, u_m$ and $\beta_0^m, \beta_1^m, \ldots, \beta_m^m$ defined as follows: u_m is some node u' satisfying Equation 30, while $\beta_i^m = (1-\beta_m^m)\beta_i^{m-1}$ for all $i = 1, \ldots, m-1$, for some β_m^m to be chosen later. One can verify that the resulting estimator $p_m(v)$ has the following inner and frontier coefficients:

$$c_m(u_i) = (1 - \beta_m^m)c_{m-1}(u_i) \qquad \forall i = 1, \ldots, m \tag{33}$$

$$c_m(u) = (1 - \beta_m^m)c_{m-1}(u) \; + \; \beta_m^m \frac{c_{G_m}(u)}{out(u)} \qquad \forall u : (u, w) \in F(G_m) \tag{34}$$

and note that by construction $c_m(u_m) = c_{m-1}(u')$. By inductive hypothesis, in $p_{m-1}(v)$ the

inner coefficients and the frontier coefficient of u_m are all identical; and therefore so are the inner coefficients of $p_{m-1}(v)$, and Equation 28 is again satisfied. Now, setting $\beta_m^m = 0$ gives $c_m(u_i) = c_{m-1}(u_i)$ and $c_m(u) = c_{m-1}(u)$, and thus by inductive hypothesis $c_m(u_i) \geq c_m(u)$ for all $i = 1, \ldots, m-1$ and all $u : (u, w) \in F(G_m)$. On the other hand, setting $\beta_m^m = 1$ gives $c_m(u_i) = 0 < c_m(u)$. We again conclude that there exists a value of β_m^m in $[0, 1)$ which makes the coefficients simultaneously satisfy also equations 29 and 30.

Finally, note that to perform the inductive step we only need to make one *Neighbourhood(·)* query to the node $u_m = u'$ in order to learn $F(G_m)$ and choose the value of β_m^m. We thus need at most $m + 1$ queries to complete the construction, including an initial query to $u_0 = v$.

A.9 Proof of Lemma 10

Consider drawing k nodes independently and uniformly at random from G, and for each u in G let ψ_u^j be the indicator random variable of the event that u is a parent of the j-th node drawn. Let then $\widehat{out}(u) = \frac{n}{k}(b\log(n) + \sum_{j=1}^k \psi_u^j)$, for any $b > 0$. Note that $\sum_{j=1}^k \psi_u^j$ is a sum of independent indicator random variables, and that $\mathbb{E}[\widehat{out}(u)] = \frac{n}{k} b\log(n) + out(u) > out(u)$. Note also that we can take a sample of ψ_u^j *simultaneously for all u in G* with an invocation of *RandomNode()* followed by an invocation of *Neighbourhood(·)* on the node returned, and thus we can take a sample of $\widehat{out}(u)$ simultaneously for all u in G with k *RandomNode()* and k *Neighbourhood(·)* queries. We prove our probabilistic bound on γ by showing probabilistic lower and upper bounds on the quantity $\frac{\widehat{out}(u)}{out(u)}$, for all u with $out(u) > 0$.

Let us start with the lower bounds, distinguishing the cases $out(u) \leq \frac{n}{k} b\log(n)$ and $out(u) > \frac{n}{k} b\log(n)$. The first case just implies $\frac{\widehat{out}(u)}{out(u)} \geq 1$. In the second case, it is easy to see that $\mathbb{E}[\sum_{j=1}^k \psi_u^j] > b\log(n)$; hence the probability that $\sum_{j=1}^k \psi_u^j$ fails to reach half its expectation is, by the probability bounds of Appendix A.1, less than $e^{-\frac{b\log(n)}{8}} = n^{-\frac{b}{8\ln(2)}}$. This is also a bound on the probability that $\widehat{out}(u) < \frac{1}{2}\mathbb{E}[\widehat{out}(u)]$ by construction of $\widehat{out}(u)$, and since $out(u) < \mathbb{E}[\widehat{out}(u)]$, on the probability that $\widehat{out}(u) < \frac{out(u)}{2}$. Taking a union bound on all u, the probability that $\frac{\widehat{out}(u)}{out(u)} < \frac{1}{2}$ for some u is at most $n^{-\frac{b}{8\ln(2)}+1}$.

Let us now turn to the upper bounds. Consider the event that $\frac{\widehat{out}(u)}{out(u)} > 2\frac{n}{k} b\log(n)$; since $\widehat{out}(u) \leq \frac{n}{k} b\log(n) out(u) + \frac{n}{k}\sum_{j=1}^k \psi_u^j$, that event implies $\sum_{j=1}^k \psi_u^j > b\log(n) out(u)$. We bound the probability of this latter event using again the bounds of Appendix A.1, setting ϵ so that $(1+\epsilon)\mathbb{E}[\sum_{j=1}^k \psi_u^j] = b\log(n) out(u)$. This requires $\epsilon \geq 1$, which implies $\frac{\epsilon^2}{2+\epsilon} \geq \frac{\frac{\epsilon+1}{2}\epsilon}{3\epsilon} = \frac{1+\epsilon}{6}$; hence at the exponent we can just plug $\frac{b\log(n) out(u)}{6}$ in place of $\frac{\epsilon^2}{2+\epsilon}\mathbb{E}[\sum_{j=1}^k \psi_u^j]$, obtaining an upper bound of $e^{-\frac{b\log(n)\cdot out(u)}{6}} \leq n^{-\frac{b}{6\ln(2)}}$. Taking a union bound on all u, the probability that $\frac{\widehat{out}(u)}{out(u)} > 2\frac{n}{k} b\log(n)$ for some u is at most $n^{-\frac{b}{6\ln(2)}+1}$.

Putting the bounds together, the probability that $\frac{\widehat{out}(u)}{out(u)} / \frac{\widehat{out}(u')}{out(u')}$ exceeds $2\frac{n}{k} b\log(n) / \frac{1}{2} = 4\frac{n}{k} b\log(n)$ for some u, u' with positive outdegrees is at most $n^{-\frac{b}{8\ln(2)}+1} + n^{-\frac{b}{6\ln(2)}+1} < 2n^{-\frac{b}{8\ln(2)}+1}$.

A.10 Dealing with dangling nodes

Essentially, if G contains dangling nodes then the estimators for $P(v)$ developed in Section 3 must be adapted by adding a random term (and, possibly, rescaling by a constant) which can be estimated with sufficient accuracy through just $O(1)$ calls to *SampleNode()*.

We start by generalizing Lemma 5 to allow for dangling nodes in G, but not in $\bar{G} \setminus \{v\}$. This is the most general assumption we need, since the subgraph $\bar{G}$ built by our algorithm never contains dangling nodes except possibly v.

Lemma 14. *Given a graph G, a node $v \in G$, and an induced subgraph $\bar{G}$ of G containing v, if there are no dangling nodes in $\bar{G} \setminus \{v\}$ then the PageRank score of v can be written as:*

$$P(v) = \mu_v^{\bar{G}} \cdot \Big(\sum_{w \in \bar{G}} \Big(\frac{1-\alpha}{n} + \sum_{\substack{u \in G \setminus \{v\} \\ out(u)=0}} P(u) \frac{\alpha}{n} \Big) \mho_{\bar{G}}(w, v) + \sum_{(u,w) \in F(\bar{G})} P(u) \frac{\alpha}{out(u)} \mho_{\bar{G}}(w, v) \Big) \tag{35}$$

where $\mu_v^{\bar{G}}$ equals $(1 - \frac{\alpha}{n} \sum_{w \in \bar{G}} \mho_{\bar{G}}(w, v))^{-1}$ if v is a dangling node and equals 1 otherwise.

Proof. We extend the proof of Lemma 5 (see Appendix A.4). For the sake of the analysis we imagine that, if the current node is dangling, with probability $1 - \alpha$ the walk performs the usual random jump and with probability α it follows one at random of n outgoing virtual arcs.

Consider first the case $out(v) > 0$, i.e. v is not a dangling node. The only modification needed is in the analysis of $\mathcal{E}_v^{\notin \bar{G}}$: now this event can take place also if the walk enters $\bar{G}$ from a dangling node of $G \setminus \bar{G}$, by following a virtual arc. If this happens, for some $k \geq 0$ the walk was $k+1$ steps ago in a dangling node u of $G \setminus \bar{G}$, then followed an outgoing virtual arc of u to a node $w \in \bar{G}$, and finally followed a path π in $\bar{G}$ of length k from w to v. The probability of being in u is $P(u)$ at any instant, the probability of following a virtual arc to a node w is $\frac{\alpha}{n}$ for any given w, and the probability of following any given path π is $\mho^\pi$. The expression of $P(\mathcal{E}_v^{\notin \bar{G}})$ given by Equation 27 thus gains a term relative to the dangling nodes in $G \setminus \bar{G}$:

$$P(\mathcal{E}_v^{\notin \bar{G}}) = \sum_{(u,w) \in F(\bar{G})} P(u) \frac{\alpha}{out(u)} \mho_{\bar{G}}(w, v) \; + \sum_{\substack{u \in G \setminus \bar{G} \\ out(u)=0}} P(u) \frac{\alpha}{n} \sum_{w \in \bar{G}} \mho_{\bar{G}}(w, v) \tag{36}$$

Since $P(v) = P(\mathcal{E}_v^{\in \bar{G}}) + P(\mathcal{E}_v^{\notin \bar{G}})$, we obtain:

$$P(v) = \sum_{w \in \bar{G}} \Big(\frac{1-\alpha}{n} + \sum_{\substack{u \in G \setminus \{v\} \\ out(u)=0}} P(u) \frac{\alpha}{n} \Big) \mho_{\bar{G}}(w, v) \; + \sum_{(u,w) \in F(\bar{G})} P(u) \frac{\alpha}{out(u)} \mho_{\bar{G}}(w, v) \tag{37}$$

Let us now see how the above expression changes when $out(v) = 0$. In this case, v has outgoing virtual arcs that the random walk can take. The analysis already includes the event that the walk did not follow any of those arcs after the last random jump, as well as the event that the last time the walk followed one of those arcs it arrived in $G \setminus \bar{G}$. We must then just add the event that, for some $k \geq 0$, the random walk $k+1$ steps ago was in v, then took one of v's outgoing virtual arcs towards a node $w \in \bar{G}$, and finally followed a path π in $\bar{G}$ of length k from w to v. The probability of being in v is $P(v)$ at any instant, the probability of taking a virtual arc to w is $\frac{\alpha}{n}$ for any node w, and the probability of taking a path π is $\mho^\pi$ for any π. Therefore when $out(v) = 0$ we obtain $P(v)$ by adding to Equation 37 the following term:

$$\sum_{k \geq 0} P(v) \sum_{w \in \bar{G}} \frac{\alpha}{n} \sum_{\substack{\pi \in \Pi_{\bar{G}}(w,v) \\ |\pi| = k}} \mho^\pi \tag{38}$$

This term equals $P(v) \frac{\alpha}{n} \sum_{w \in \bar{G}} \mho_{\bar{G}}(w, v)$, and thus $P(v)$ is given by the expression of Equation 37 multiplied by $(1 - \frac{\alpha}{n} \sum_{w \in \bar{G}} \mho_{\bar{G}}(w, v))^{-1}$. We can then define a constant $\mu_v^{\bar{G}}$ equalling 1 if $out(v) > 0$ and $(1 - \frac{\alpha}{n} \sum_{w \in \bar{G}} \mho_{\bar{G}}(w, v))^{-1}$ if $out(v) = 0$, concluding the proof. □

We can now adapt the estimators of $P(v)$. Define $P_\emptyset = \sum_{u \in G \setminus \{v\} : out(u) = 0} P(u)$, the sum of the PageRank scores of all dangling nodes in $G \setminus \bar{G}$. Starting from Equation 35, it is easy to verify that Equation 3 becomes:

$$P(v) = \big(1 + \frac{\alpha}{1-\alpha} P_\emptyset\big) \mu_v^{\bar{G}} c_{\bar{G}} \; + \; \mu_v^{\bar{G}} \sum_{u : (u,w) \in F(\bar{G})} P(u) \cdot \frac{c_{\bar{G}}(u)}{out(u)} \tag{39}$$

These modifications also apply to the estimator $p_{\bar{G}}(v)$ given by Equation 4 in Lemma 6, and then propagate through Equations 5, 6, 7, 8 and 9. The resulting estimators are thus just a rescaled version of the original ones, with a first term further multiplied by $(1 + \frac{\alpha}{1-\alpha} P_\emptyset)$. If we are able to accurately estimate $(1 + \frac{\alpha}{1-\alpha} P_\emptyset)$ with few queries, our final results still hold.

Consider then $p_\emptyset = \frac{1}{\ell} \sum_{h=1}^{\ell} \sum_{u \in G \setminus \{v\} : out(u)=0} \chi_u^h$, the average of ℓ independent indicator random variables of the event that *SampleNode()* returns a dangling node in $G \setminus \{v\}$. Clearly $\mathbb{E}[p_\emptyset] = P_\emptyset$, and we can use $(1 + \frac{\alpha}{1-\alpha} p_\emptyset)$ as an estimator of $(1 + \frac{\alpha}{1-\alpha} P_\emptyset)$. It is easy to check that, if $(1 + \frac{\alpha}{1-\alpha} p_\emptyset)$ falls off its expectation by more than a factor $1 \pm \epsilon$, then $\ell\, p_\emptyset$ falls off its expectation by more than a factor $1 \pm \epsilon\big(1 + \frac{1-\alpha}{\alpha P_\emptyset}\big)$. Since $\ell\, p_\emptyset$ is a sum of non-positively correlated indicator random variables with expectation $\ell\, P_\emptyset$, by the probability bounds of Appendix A.1 the probability that such an event takes place is at most

$$2e^{-\ell\, P_\emptyset\, \epsilon^2 (1 + \frac{1-\alpha}{\alpha P_\emptyset})^2/3} < 2e^{-\ell\, P_\emptyset\, \epsilon^2 (\frac{1-\alpha}{\alpha P_\emptyset})^2/3} = 2e^{-\frac{\ell \epsilon^2}{P_\emptyset} (\frac{1-\alpha}{\alpha})^2/3} < 2e^{-\ell \epsilon^2 (\frac{1-\alpha}{\alpha})^2/3} \tag{40}$$

where the last inequality follows from the fact that $P_\emptyset < 1$ by definition of PageRank score. Since for any $\delta > 0$ we have $2e^{-\ell \epsilon^2 (\frac{1-\alpha}{\alpha})^2/3} \leq \delta$ when $\ell = \frac{3\alpha^2}{\epsilon^2 (1-\alpha)^2} \ln(\frac{2}{\delta})$, by a simple union bound the results of Theorems 1 and 2 remain valid by paying an additional $O\big(\frac{1}{\epsilon^2} \ln(\frac{2}{\delta})\big)$ queries.

A.11 Porting our techniques outside PageRank

Although developed for the specific case of PageRank, our techniques can be abstracted and may prove useful in designing low-query-complexity algorithms for other local approximation problems. In particular, the technique for building a balanced estimator of $P(v)$ starting from the diffuse formulation of Lemma 5 can be readily generalized; it actually allows one to build a balanced estimator for any nonnegative quantity $X(v)$ that, given a subgraph $\bar{G}$ of G, admits a diffuse formulation in the form

$$X(v) = f_{\bar{G}}(v) + \sum_{(u,w) \in F(\bar{G})} X(u) \cdot g_{\bar{G}}(w, v) \tag{41}$$

as long as $f_{\bar{G}}(v)$ and $g_{\bar{G}}(w, v)$ can be computed from $\bar{G}$, and if one can sample random variables χ_u with expectation $X(u)$. We give two representative examples where this technique can be applied immediately. In both cases, investigating the precise query complexity bounds that arise from the interplay of the various parameters involved is an open direction of research.

Katz's centrality. In a graph G, the Katz score of node v is $S(v) = \sum_{i=0}^{\infty} C_i(v) \beta^i$, where $C_i(v)$ is the number of paths of length i ending in v (an arc may appear multiple times), and $\beta > 0$ is small enough to guarantee convergence. It is easy to see that a node's score can be rewritten in terms of the scores of its neighbours, and thus we can build a balanced estimator for $S(v)$ using our technique under the same exploration model used in this paper. The tricky part is defining random variables χ_u (not necessarily indicator) that have expectation proportional to $S(u)$, which may be done in the following way. Consider a random walk on G that, starting from a node at random, at each step either takes an outgoing arc at random or terminates with probability $1 - \beta$; it can be shown that $S(u)$ is proportional to the probability that the walk terminates in u times the product of the outdegrees of the nodes the walk visited before u. It then suffices to define χ_u to equal the product of those outdegrees if the walk terminates in u, or 0 otherwise.

Markov chains. A remarkably general example is the estimation of single-state stationary probabilities in ergodic Markov chains. Here G represents the transition probability matrix of the chain, and querying a state u reveals the corresponding row and column. It is easy to see that one can express the stationary probability $\pi(u)$ of u in terms of stationary probabilities of the

neighboring states, and therefore we can build a balanced estimator of $\pi(u)$ using our technique. One can sample the estimator by sampling from the stationary distribution of the chain, e.g. through simulation – roughly τ queries would suffice to take one sample if the chain has mixing time τ. The variance of the estimator depends on the aggregate stationary probability of all states involved in the estimator, in the same way as above the variance of $p_m^\ell(v)$ depended on $\mathbb{E}[\frac{\ell}{c}(p_m^\ell(v) - c_m)]$: the largest the latter, the lower the former.

A.12 Proof of Theorem 3

The proof hinges on the construction of a graph G of arbitrarily large size n, formed by two nearly identical subgraphs that one must distinguish in order to decide which of u and v has the higher score. The bound on the number of queries is obtained in expectation assuming that the decision must be made on a graph chosen uniformly at random from the family of all graphs isomorphic to G, and thus holds for at least one of those graphs. The two subgraphs (see Figure 1) contain a first level of ancestors whose contribution is "damped" by the presence of additional children (allowing one to increase the number of those ancestors while leaving the scores of u and v essentially unchanged), and differ solely for the presence, in one of the two, of a second level of ancestors which ensures that the scores of u and v are not within a factor $(1+\eta)$ of each other.

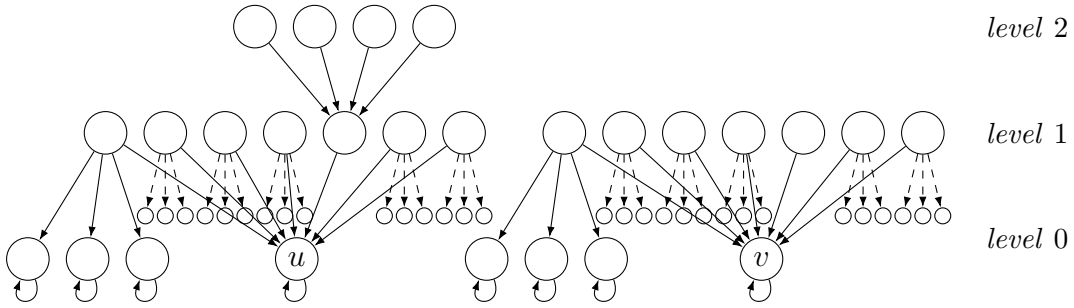

Figure 1: The structure of the generic graph G used in the proof. To decide which of u and v has the higher score, one must distinguish the two nearly-identical subgraphs.

To distinguish the two subgraphs, one must at least discover some second-level node, and this can only be done by either exploring the first-level nodes via local queries (starting from u and v themselves) or invoking global queries that may directly lead to a second-level node. With a careful choice of the number of nodes in each level, and with an adaptation that keeps the construction working for "small" $f(n)$, one can always ensure a maximum outdegree $O(\gamma(n))$, scores of u and v in $\Theta(f(n))$ but not within a factor $(1+\eta)$ of each other, and an expected $\Omega\big(\min\big(\frac{1}{f(n)}, n^{\frac{2}{3}}, n^{\frac{1}{2}}\gamma(n)^{\frac{1}{2}}\big)\big)$ queries to find any second-level node and thus to determine which of u and v has the higher score with probability $\frac{1}{2} + \Omega(1)$ – proving the theorem.

Let us now proceed to the formal proof. We analyse separately, but with similar techniques, the two cases $\gamma(n) > n^{\frac{1}{3}}$ and $1 \leq \gamma(n) \leq n^{\frac{1}{3}}$, assembling the general lower bound at the end. We will use the following equivalent formulation of PageRank (see e.g. [6]), which is easier to compute explicitly on the graphs appearing in this proof:

$$P(v) = \frac{1-\alpha}{n} \sum_{\tau=0}^{+\infty} \alpha^\tau \Big(\sum_{z \in G} \Big(\sum_{\pi_{z,v}:|\pi_{z,v}|=\tau} \Big(\prod_{(w,w') \in \pi_{z,v}} \frac{1}{out(w)} \Big) \Big) \Big) \tag{42}$$

where a path $\pi_{z,v}$ is a sequence of zero or more arcs leading from z to v, and $|\pi_{z,v}|$ is the number of such arcs. In the case $out(v) = 1$ and its arc forms a self loop, we simplify the computation of $P(v)$ as in [13] by disregarding that self loop (and thus all the paths it belongs to), applying Equation 42, and then multiplying the result by $\frac{1}{1-\alpha}$. To keep the proof simple, we assume that

$f(n)$ and $\gamma(n)$ are respectively in $\Theta(f(n_0))$ and $\Theta(\gamma(n_0))$ whenever $n = \Theta(n_0)$, and show how to remove this hypothesis at the end.

Case $\gamma(n) > n^{\frac{1}{3}}$. We proceed as follows: first, for any $n^{-2/3} \leq f(n) \leq 1$ we exhibit a family $\mathcal{F}_n$ of arbitrarily large, isomorphic graphs on n nodes whose generic element has maximum outdegree $O(\gamma(n))$ and contains two nodes u, v with PageRank scores in $\Theta(f(n))$ but not within a factor $(1+\eta)$ of each other. We prove that any algorithm, on some element in $\mathcal{F}_n$, must perform $\Omega\big(\frac{1}{f(n)}\big)$ queries in expectation to decide which among u and v has the higher score with probability $\frac{1}{2} + \Omega(1)$. We then show how to adapt $\mathcal{F}_n$ for $\frac{1}{n} \leq f(n) < n^{-2/3}$ while bringing the above expectation to $\Omega(n^{2/3})$, thus obtaining a lower bound of $\Omega(\min(\frac{1}{f(n)}, n^{2/3}))$ for any $\frac{1}{n} \leq f(n) \leq 1$.

To build a generic element G of $\mathcal{F}_n$ for $n^{-2/3} \leq f(n) \leq 1$, consider an arbitrary positive integer n_0 (a rough approximation of the final number of nodes in G). The nodes of G (see Figure 2) can be divided into three levels. Level 0 consists of 2 nodes u, v and $2\left\lceil \frac{n_0\sqrt{f(n_0)}}{\alpha} \right\rceil \left\lceil \frac{1}{\sqrt{f(n_0)}} \right\rceil$ nodes $w_0^0, w_0^1, \ldots$; each node in this level has (only) a self loop. Level 1 consists of 2 nodes s_u, s_v and $2\left\lceil \frac{n_0\sqrt{f(n_0)}}{\alpha} \right\rceil$ nodes $w_1^0, w_1^1, \ldots$. The sole outgoing arcs of s_u and s_v are, respectively, (s_u, u) and (s_v, v); while (w_1^j, u) is an arc for $0 \leq j < \left\lceil \frac{n_0\sqrt{f(n_0)}}{\alpha} \right\rceil$, and (w_1^j, v) is an arc for $\left\lceil \frac{n_0\sqrt{f(n_0)}}{\alpha} \right\rceil \leq j < 2\left\lceil \frac{n_0\sqrt{f(n_0)}}{\alpha} \right\rceil$. Also, each node w_1^j has outgoing arcs $(w_1^j, w_0^{\lceil 1/\sqrt{f(n_0)} \rceil j}), \ldots, (w_1^j, w_0^{\lceil 1/\sqrt{f(n_0)} \rceil (j+1)-1})$. Finally, level 2 consists of $\left\lceil \frac{c\eta n_0 f(n_0)}{\alpha^2} \right\rceil$ nodes $w_2^0, w_2^1, \ldots$ having outgoing arcs $(w_2^0, s_u), (w_2^1, s_u), \ldots$, towards s_u, for a constant $c > 0$ to be determined later. The family $\mathcal{F}_n$ consists of all the graphs isomorphic to G. In particular, in an element drawn uniformly at random from $\mathcal{F}_n$, each node is identified by a random integer in $\{1, \ldots, n\}$; but we will keep denoting by u and v the (only) two level-0 nodes each having $1 + \left\lceil \frac{n_0\sqrt{f(n_0)}}{\alpha} \right\rceil \geq 2$ incoming arcs from level-1 nodes, and by s_u and s_v the (only) two level-1 nodes whose sole arc points towards u and v respectively, so that s_u is the node pointed by all the level-2 nodes.

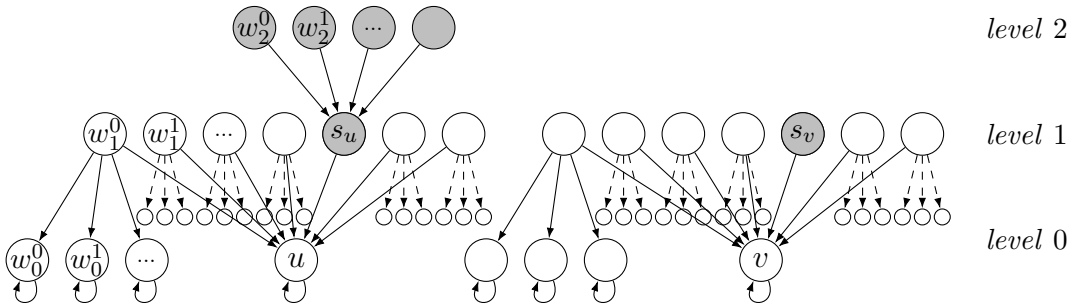

Figure 2: A generic graph G of the family $\mathcal{F}_n$ for $\gamma(n) > n^{\frac{1}{3}}$ and $n^{-2/3} \leq f(n) \leq 1$. All level-1 nodes except s_u and s_v have $\left\lceil \frac{1}{\sqrt{f(n_0)}} \right\rceil$ additional children besides u or v but, for simplicity, we have drawn these children in normal size only for the leftmost node. In order to distinguish the two subgraphs, and thus u from v, one must find at least one of the nodes depicted in grey.

The number of nodes in G is $n = 4 + 2\left\lceil \frac{n_0\sqrt{f(n_0)}}{\alpha} \right\rceil \left\lceil \frac{1}{\sqrt{f(n_0)}} \right\rceil + 2\left\lceil \frac{n_0\sqrt{f(n_0)}}{\alpha} \right\rceil + \left\lceil \frac{c\eta n_0 f(n_0)}{\alpha^2} \right\rceil$, which is a $\Theta(n_0)$ between $\frac{2n_0}{\alpha}$ and $\frac{17+c\eta}{\alpha^2} n_0$; therefore, one can make G arbitrarily large by increasing n_0. The maximum outdegree of the graph is $\left\lceil \frac{1}{\sqrt{f(n_0)}} \right\rceil = \Theta\big(\frac{1}{\sqrt{f(n)}}\big)$, and since $f(n) \geq n^{-\frac{2}{3}}$, this

is an $O(n^{\frac{1}{3}}) \subseteq O(\gamma(n))$. The PageRank scores of u and v are:

$$P(u) = \frac{1}{n}\left(1 + \alpha\left(1 + \frac{\left\lceil \frac{n_0\sqrt{f(n_0)}}{\alpha} \right\rceil}{1 + \left\lceil \frac{1}{\sqrt{f(n_0)}} \right\rceil}\right) + \alpha^2 \left\lceil \frac{c\eta n_0 f(n_0)}{\alpha^2} \right\rceil\right) \tag{43}$$

$$P(v) = \frac{1}{n}\left(1 + \alpha\left(1 + \frac{\left\lceil \frac{n_0\sqrt{f(n_0)}}{\alpha} \right\rceil}{1 + \left\lceil \frac{1}{\sqrt{f(n_0)}} \right\rceil}\right)\right) \tag{44}$$

and both are in $\Theta(\frac{n_0 f(n_0)}{n})$, and since $n = \Theta(n_0)$, also both in $\Theta(f(n))$. Their difference $P(u) - P(v)$ is equal to $\frac{1}{n}\alpha^2 \left\lceil \frac{c\eta n_0 f(n_0)}{\alpha^2} \right\rceil$, and therefore:

$$\frac{P(u) - P(v)}{P(v)} = \frac{\frac{1}{n}\alpha^2 \left\lceil \frac{c\eta n_0 f(n_0)}{\alpha^2} \right\rceil}{\frac{1}{n}\left(1 + \alpha\left(1 + \frac{\left\lceil \frac{n_0\sqrt{f(n_0)}}{\alpha} \right\rceil}{1 + \left\lceil \frac{1}{\sqrt{f(n_0)}} \right\rceil}\right)\right)} \geq \frac{\frac{1}{n} c\eta n_0 f(n_0)}{\frac{1}{n}\left(1 + \alpha + \frac{2n_0\sqrt{f(n_0)}}{\frac{1}{\sqrt{f(n_0)}}}\right)} = \frac{c\eta}{\frac{1+\alpha}{n_0 f(n_0)} + 2} \tag{45}$$

and for any $c \geq 4$ the last term is greater than η for any $\eta > 0$, any $\alpha \in (0, 1)$, and any $n_0 > 0$.

Consider now a generic (Monte Carlo) algorithm that, using only natural exploration queries, must decide which among u and v has the higher PageRank score in a graph drawn uniformly at random from $\mathcal{F}_n$ (the ids of u and v are given in input in random order). Note that the information returned by global queries is independent from their order relative to local queries, since the output of a global query does not depend on the output of previous local queries. We can then simplify the analysis by ideally grouping together all the global queries in a first phase, followed by all the local queries in a second phase, with each phase sporting at most q queries if the algorithm performs at most q queries overall.

Let us consider the first phase. We reinforce the algorithm with an oracle that operates as follows: when a global query outputs the id of a node among s_u, s_v or $w_2^0, w_2^1, \ldots$, it lets the algorithm immediately return the correct ranking of the input nodes; and when a global query outputs the id of a level-0 node not in $\{u, v\}$, it reveals both the id of its unique parent (which can thus be marked as neither s_u nor s_v) and the ids of all its siblings (revealing all the information that would be given by a *Neighbourhood(·)* on their parent). Since the graph is drawn uniformly at random from $\mathcal{F}_n$, each single global query returns the id of a node among s_u, s_v or $w_2^0, w_2^1, \ldots$ with probability $\left(2 + \left\lceil \frac{c\eta n_0 f(n_0)}{\alpha^2} \right\rceil\right)/n = O(f(n))$; hence, the probability that some global query in the first phase returns such an id is $O(q \cdot f(n))$. If this does not happen, then the algorithm possesses no information about which one of the input nodes has the higher PageRank score: conditioned on the fact of *not* discovering any of the above nodes, each of the possible output sequences of the global queries has exactly the same probability.

Let us turn to the second phase, conditioned on the event that the first phase ended with the algorithm having no information about which input node has the higher score. Note that a single invocation of *Neighbourhood(·)* yields sufficient information for the algorithm to emulate, without any further query invocation, any other type of local query; we can then assume that the algorithm employs only *Neighbourhood(·)* queries, and slightly reinforce it by allowing two *Neighbourhood(·)* invocations on the two input nodes at no cost before the phase begins. At this point, the algorithm knows the ids of all level-1 nodes; and, possibly, the ids of some level-0 nodes discovered during the first phase, together with the corresponding ids of their $\leq q$ parents marked as neither s_u nor s_v. We can then assume that the algorithm invokes *Neighbourhood(·)* on the ids of unmarked level-1 nodes, as querying any other node would not yield any information not already possessed; and we can reinforce it with an oracle that lets it immediately return the correct ranking of the input nodes upon querying s_u or s_v. At any point, at most q level-1 nodes have been marked as neither s_u nor s_v by the at most q local and/or global queries; and thus,

since the graph is drawn uniformly at random from $\mathcal{F}_n$, at any point the probability of querying s_u or s_v is at most $1/\big(1+\left\lceil\frac{n_0\sqrt{f(n_0)}}{\alpha}\right\rceil-q\big)$. Therefore, the overall probability of querying s_u or s_v in the second phase is at most $q/\big(1+\left\lceil\frac{n_0\sqrt{f(n_0)}}{\alpha}\right\rceil-q\big)$, which for $q<\frac{1}{2}\left\lceil\frac{n_0\sqrt{f(n_0)}}{\alpha}\right\rceil$ is in $O\big(\frac{q}{n\sqrt{f(n)}}\big)$, which for any $f(n)\geq n^{-2/3}$ is in $O(q\cdot f(n))$. If neither s_u nor s_v are found, the algorithm has once again no information about which input node has the higher PageRank score; therefore, conditioned on the event of reaching the end of the second phase, the probability that the algorithm returns the correct ranking is $\frac{1}{2}$. Since the probability of stopping before the end of the second phase and returning the correct ranking is in $O(q\cdot f(n))$ for any $q<\frac{1}{2}\left\lceil\frac{n_0\sqrt{f(n_0)}}{\alpha}\right\rceil$, we thus have a total probability bound of $\frac{1}{2}+O(q\cdot f(n))$ on the event of returning the correct ranking for any $q<\frac{1}{2}\left\lceil\frac{n_0\sqrt{f(n_0)}}{\alpha}\right\rceil$.

Considering the uniform distribution over the elements of $\mathcal{F}_n$, denote by $\mu=\mu(\mathcal{F}_n)$ the expected number of queries employed by the algorithm, by $p_q=p_q(\mathcal{F}_n)$ the probability that it employs more than q queries, and by $Pr[success]$ the probability that it returns the correct ranking of the input nodes. $Pr[success]$ is then upper bounded by $p_q\cdot 1$ (at best, the algorithm always returns the correct ranking when using more than q queries) plus $(1-p_q)\leq 1$ times the probability of returning the correct ranking on an element drawn uniformly at random from $\mathcal{F}_n$ using at most q queries. Using Markov's inequality to write $p_q<\frac{\mu}{q}$, for any $q<\frac{1}{2}\left\lceil\frac{n_0\sqrt{f(n_0)}}{\alpha}\right\rceil$ the following holds:

$$Pr[success]\leq\frac{\mu}{q}\cdot 1+1\cdot\Big(\frac{1}{2}+O(q\cdot f(n))\Big)=\frac{1}{2}+O\Big(\frac{\mu}{q}+q\cdot f(n)\Big) \tag{46}$$

For any $\mu\in o\big(\frac{1}{f(n)}\big)$ we can thus pick some $q(n)\in o\big(\frac{1}{f(n)}\big)\cap\ \omega(\mu(\mathcal{F}_n))$ and an n_0 sufficiently large to ensure $q(n)<\frac{1}{2}\left\lceil\frac{n_0\sqrt{f(n_0)}}{\alpha}\right\rceil$, and Equation 46 yields $Pr[success]=\frac{1}{2}+o(1)$. This holds on average over the elements of $\mathcal{F}_n$ for any algorithm; and thus, for every given algorithm there exists some $G\in\mathcal{F}_n$ on which that algorithm must perform an expected $\Omega\big(\frac{1}{f(n)}\big)$ queries to return the correct ranking with probability $\frac{1}{2}+\Omega(1)$.

For $\frac{1}{n}\leq f(n)<n^{-\frac{2}{3}}$, we obtain the generic element of $\mathcal{F}_n$ by adapting that built for $f(n)=n^{-\frac{2}{3}}$. First, remove the self-loops (u,u) and (v,v). Then, let $k=\left\lceil\log_\alpha\left(f(n_0)\cdot n_0^{2/3}\right)\right\rceil>0$, and add $2k$ nodes $z_u^1,\dots,z_u^k$ and $z_v^1,\dots,z_v^k$, and $2k$ arcs $(u,z_u^1),(z_u^1,z_u^2),\dots,(z_u^{k-1},z_u^k)$ and $(v,z_v^1),(z_v^1,z_v^2),\dots,(z_v^{k-1},z_v^k)$; and add two self-loops (z_u^k,z_u^k), (z_v^k,z_v^k). The size of the graph is clearly still $\Theta(n_0)$ and can be made arbitrarily large. The maximum outdegree of the graph is $\Theta(n^{\frac{1}{3}})=O(\gamma(n))$. The PageRank scores of z_u^k and z_v^k are:

$$P(z_u^k)=\frac{1}{n}\left(\sum_{i=0}^{k-1}\alpha^i+\alpha^k\left(1+\alpha\left(1+\frac{\left\lceil\frac{n_0^{2/3}}{\alpha}\right\rceil}{1+\left\lceil n_0^{1/3}\right\rceil}\right)+\alpha^2\left\lceil\frac{c\eta n_0^{1/3}}{\alpha^2}\right\rceil\right)\right) \tag{47}$$

$$P(z_v^k)=\frac{1}{n}\left(\sum_{i=0}^{k-1}\alpha^i+\alpha^k\left(1+\alpha\left(1+\frac{\left\lceil\frac{n_0^{2/3}}{\alpha}\right\rceil}{1+\left\lceil n_0^{1/3}\right\rceil}\right)\right)\right) \tag{48}$$

and thus both in $\Theta\big(\frac{\alpha^k n_0^{1/3}}{n}\big)$, and since $\alpha^k=\Theta(f(n_0)\cdot n_0^{2/3})$, both in $\Theta\big(\frac{f(n_0)\cdot n_0}{n}\big)=\Theta(f(n))$.

Their difference $P(z_u^k) - P(z_v^k)$ is equal to $\frac{1}{n}\alpha^{k+2}\left\lceil \frac{c\eta n_0^{1/3}}{\alpha^2} \right\rceil$, and therefore

$$\frac{P(z_u^k) - P(z_v^k)}{P(z_v^k)} = \frac{\frac{1}{n}\alpha^{k+2}\left\lceil \frac{c\eta n_0^{1/3}}{\alpha^2} \right\rceil}{\frac{1}{n}\left(\sum_{i=0}^{k-1} \alpha^i + \alpha^k\left(1 + \alpha\left(1 + \frac{\left\lceil \frac{n_0^{2/3}}{\alpha} \right\rceil}{1+\left\lceil n_0^{1/3} \right\rceil}\right)\right)\right)} \geq \frac{\alpha^k c\eta n_0^{\frac{1}{3}}}{\left(\frac{1}{1-\alpha} + \alpha^k 2n_0^{\frac{1}{3}}\right)} \tag{49}$$

and by the choice of α and since $f(n_0) \geq \frac{1}{n_0}$, then $\alpha < \alpha^k n_0^{\frac{1}{3}} \leq 1$ and the last term of Equation 49 is greater than $\frac{c\alpha\eta}{1/(1-\alpha)+2\alpha}$, which is greater than or equal to η for any $c \geq 2 + \frac{1}{\alpha(1-\alpha)}$.

The family $\mathcal{F}_n$ consists again of all the graphs isomorphic to G, and we give in input to the algorithm, in random order, the ids of z_u^k and z_v^k in a graph drawn uniformly at random from $\mathcal{F}_n$. But returning the correct ranking of z_u^k and z_v^k is equivalent to returning that of u and v, therefore the $\Omega(n^{2/3})$ lower bound obtained for $f(n) = n^{-\frac{2}{3}}$ must hold; and for $f(n) < n^{-\frac{2}{3}}$, this bound is equivalent to $\Omega\big(\min\left(\frac{1}{f(n)}, n^{2/3}\right)\big)$.

We thus have a bound of $\Omega\big(\min\left(\frac{1}{f(n)}, n^{2/3}\right)\big)$ for any $\gamma(n) > n^{\frac{1}{3}}$ and $\frac{1}{n} \leq f(n) \leq 1$.

Case $1 \leq \gamma(n) \leq n^{\frac{1}{3}}$. We proceed as follows: first, for any $f(n) \geq n^{-\frac{1}{2}}\gamma(n)^{-\frac{1}{2}}$ we exhibit a family $\mathcal{F}_n$ of arbitrarily large, isomorphic graphs on n nodes whose generic element has maximum outdegree $\Theta(\gamma(n))$ and contains two nodes u, v with PageRank scores in $\Theta(f(n))$ but not within a factor $(1+\eta)$ of each other. We prove that any algorithm, on some element in $\mathcal{F}_n$, must perform $\Omega\big(\frac{1}{f(n)}\big)$ queries in expectation to decide which among u and v has the higher score with probability $\frac{1}{2} + \Omega(1)$. We then show how to adapt $\mathcal{F}_n$ for $\frac{1}{n} \leq f(n) < n^{-\frac{1}{2}}\gamma(n)^{-\frac{1}{2}}$ while bringing the above expectation to $\Omega(n^{\frac{1}{2}}\gamma(n)^{\frac{1}{2}})$, thus obtaining a lower bound of $\Omega(\min(\frac{1}{f(n)}, n^{\frac{1}{2}}\gamma(n)^{\frac{1}{2}}))$ for any $\frac{1}{n} \leq f(n) \leq 1$.

To build a generic element G of $\mathcal{F}_n$ for $f(n) \geq n^{-\frac{1}{2}}\gamma(n)^{-\frac{1}{2}}$, consider an arbitrary positive integer n_0 (a rough approximation of the final number of nodes in G). The nodes of G (see Figure 3) can be divided into three levels, plus some additional isolated nodes (see below). Level 0 consists of 2 nodes u, v and $2\left\lceil \frac{\sqrt{n_0\gamma(n_0)}}{\alpha} \right\rceil \left\lceil \gamma(n_0) \right\rceil$ nodes $w_0^0, w_0^1, \ldots$; each node in this level has (only) a self loop. Level 1 consists of 2 nodes s_u, s_v and $2\left\lceil \frac{\sqrt{n_0\gamma(n_0)}}{\alpha} \right\rceil$ nodes $w_1^0, w_1^1, \ldots$. The sole outgoing arcs of s_u and s_v are, respectively, (s_u, u) and (s_v, v); while (w_1^j, u) is an arc for $0 \leq j < \left\lceil \frac{\sqrt{n_0\gamma(n_0)}}{\alpha} \right\rceil$, and (w_1^j, v) is an arc for $\left\lceil \frac{\sqrt{n_0\gamma(n_0)}}{\alpha} \right\rceil \leq j < 2\left\lceil \frac{\sqrt{n_0\gamma(n_0)}}{\alpha} \right\rceil$. Furthermore, each node w_1^j has outgoing arcs $(w_1^j, w_0^{\lceil\gamma(n_0)\rceil j}), \ldots, (w_1^j, w_0^{\lceil\gamma(n_0)\rceil(j+1)-1})$. Finally, level 2 consists of $\left\lceil \frac{c\eta}{\alpha^2} n_0 f(n_0) \right\rceil$ nodes $w_2^0, w_2^1, \ldots$ having outgoing arcs $(w_2^0, s_u), (w_2^1, s_u), \ldots$, towards s_u, for a constant $c > 0$ to be determined later. The family $\mathcal{F}_n$ consists of all the graphs isomorphic to G. In particular, in an element drawn uniformly at random from $\mathcal{F}_n$, each node is identified by a random integer in $\{1, \ldots, n\}$; but we will keep denoting by u and v the (only) two level-0 nodes each having $1 + \left\lceil \frac{\sqrt{n_0\gamma(n_0)}}{\alpha} \right\rceil \geq 2$ incoming arcs from level-1 nodes, and by s_u and s_v the (only) two level-1 nodes whose sole arc points towards u and v respectively, so that s_u is the node pointed by all the level-2 nodes.

The number of nodes in G is $n = 4 + 2\left\lceil \frac{\sqrt{n_0\gamma(n_0)}}{\alpha} \right\rceil \left\lceil \gamma(n_0) \right\rceil + 2\left\lceil \frac{\sqrt{n_0\gamma(n_0)}}{\alpha} \right\rceil + \left\lceil \frac{c\eta}{\alpha^2} n_0 f(n_0) \right\rceil$, which is always between $4\sqrt{n_0}$ and $\frac{16+2c\eta}{\alpha^2} n_0$. We then complete G by adding isolated nodes until $n = \frac{16+2c\eta}{\alpha^2} n_0 \in \Theta(n_0)$, which one can make arbitrarily large by increasing n_0. The maximum

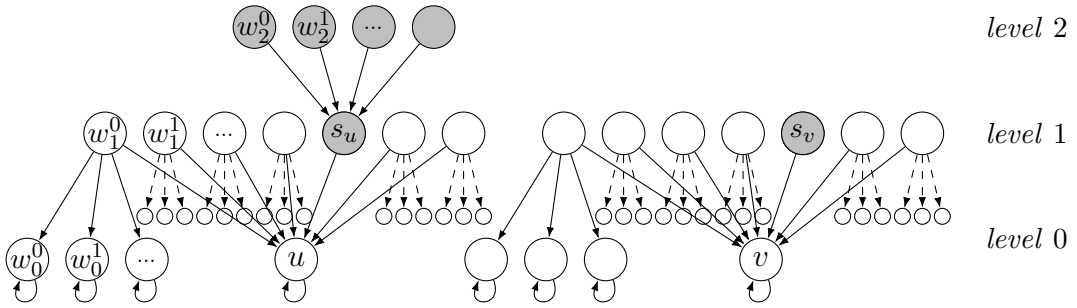

Figure 3: A generic graph G of the family $\mathcal{F}_n$ for $1 \leq \gamma(n) \leq n^{\frac{1}{3}}$ and $f(n) \geq n^{-\frac{1}{2}}\gamma(n)^{-\frac{1}{2}}$. All level-1 nodes except s_u and s_v have $\lceil \gamma(n_0) \rceil$ additional children besides u or v but, for simplicity, we have drawn these children in normal size only for the leftmost node. Again, in order to distinguish the two subgraphs, and thus u from v, one must find at least one of the nodes depicted in grey.

outdegree of the graph is $1 + \lceil \gamma(n_0) \rceil = \Theta(\gamma(n))$. The PageRank scores of u and v are:

$$P(u) = \frac{1}{n}\left(1 + \alpha\left(1 + \frac{\left\lceil \frac{\sqrt{n_0\gamma(n_0)}}{\alpha} \right\rceil}{1 + \lceil \gamma(n_0) \rceil}\right) + \alpha^2 \left\lceil \frac{c\eta}{\alpha^2} n_0 f(n_0) \right\rceil\right) \tag{50}$$

$$P(v) = \frac{1}{n}\left(1 + \alpha\left(1 + \frac{\left\lceil \frac{\sqrt{n_0\gamma(n_0)}}{\alpha} \right\rceil}{1 + \lceil \gamma(n_0) \rceil}\right)\right) \tag{51}$$

and both are in $\Theta\big(\frac{1}{n}\big(\sqrt{\frac{n_0}{\gamma(n_0)}} + n_0 f(n_0)\big)\big)$, and since $n = \Theta(n_0)$ and $f(n) \geq n^{-\frac{1}{2}}\gamma(n)^{-\frac{1}{2}}$, both in $\Theta(f(n))$. Their difference $P(u) - P(v)$ is equal to $\frac{1}{n}\alpha^2 \left\lceil \frac{c\eta}{\alpha^2} n_0 f(n_0) \right\rceil$, and therefore:

$$\frac{P(u) - P(v)}{P(v)} = \frac{\frac{1}{n}\alpha^2 \left\lceil \frac{c\eta}{\alpha^2} n_0 f(n_0) \right\rceil}{\frac{1}{n}\left(1 + \alpha\left(1 + \frac{\left\lceil \frac{\sqrt{n_0\gamma(n_0)}}{\alpha} \right\rceil}{1 + \lceil \gamma(n_0) \rceil}\right)\right)} \tag{52}$$

$$\geq \frac{c\eta n_0 f(n_0)}{1 + \alpha + \frac{2\sqrt{n_0\gamma(n_0)}}{\gamma(n_0)}} \tag{53}$$

$$= \frac{c\eta}{\frac{1+\alpha}{n_0 f(n_0)} + \frac{2}{f(n_0)\sqrt{n_0\gamma(n_0)}}} \tag{54}$$

and since $f(n) \geq n^{-\frac{1}{2}}\gamma(n)^{-\frac{1}{2}} \geq \frac{1}{n}$, for $c \geq 4$ the last term is greater than η for any $\eta > 0$, any $\alpha \in (0, 1)$, and any $n_0 > 0$.

Consider now a generic algorithm that receives in input, in random order, the ids of nodes u and v in a graph drawn uniformly at random from $\mathcal{F}_n$, and must decide which one has the higher PageRank score using only natural exploration queries. We apply the same argument deployed for the case $\gamma(n) > n^{\frac{1}{3}}$ (see above), with now $\left\lceil \frac{\sqrt{n_0\gamma(n_0)}}{\alpha} \right\rceil = \Theta(n^{\frac{1}{2}}\gamma(n)^{\frac{1}{2}})$ level-1 nodes and $\left\lceil \frac{c\eta}{\alpha^2} n_0 f(n_0) \right\rceil = \Theta(nf(n))$ level-2 nodes; note that the presence of isolated nodes only reduces the probability that a global query reveals useful information, and do not influence the probability that a local query reveals useful information. This gives a lower bound of $\Omega\big(\min\big(n^{\frac{1}{2}}\gamma(n)^{\frac{1}{2}}, \frac{1}{f(n)}\big)\big)$ queries to bring the probability of success on $\mathcal{F}_n$ to $\frac{1}{2} + \Omega(1)$.

For $\frac{1}{n} \leq f(n) < n^{-\frac{1}{2}}\gamma(n)^{-\frac{1}{2}}$, we obtain the generic element of $\mathcal{F}_n$ by adapting the graph built for $f(n) = n^{-\frac{1}{2}}\gamma(n)^{-\frac{1}{2}}$. First, remove the self-loops (u, u) and (v, v). Then let $k = \left\lceil \log_\alpha \left(f(n_0) \cdot n_0^{1/2} \cdot \gamma(n_0)^{1/2}\right) \right\rceil > 0$, and add $2k$ nodes $z_u^1, \ldots, z_u^k$ and $z_v^1, \ldots, z_v^k$, and $2k$ arcs

$(u, z_u^1), (z_u^1, z_u^2), \ldots, (z_u^{k-1}, z_u^k)$ and $(v, z_v^1), (z_v^1, z_v^2), \ldots, (z_v^{k-1}, z_v^k)$; and add two self-loops (z_u^k, z_u^k), (z_v^k, z_v^k). The size of the graph is clearly still $\Theta(n_0)$ and can be made arbitrarily large. The maximum outdegree of the graph is still $\Theta(\gamma(n))$. The PageRank scores of z_u^k and z_v^k are:

$$P(z_u^k) = \frac{1}{n}\left(\sum_{i=0}^{k-1}\alpha^i + \alpha^k\left(1+\alpha\left(1+\frac{\left\lceil\frac{\sqrt{n_0\gamma(n_0)}}{\alpha}\right\rceil}{1+\lceil\gamma(n_0)\rceil}\right)+\alpha^2\left\lceil\frac{c\eta}{\alpha^2}\sqrt{\frac{n_0}{\gamma(n_0)}}\right\rceil\right)\right) \tag{55}$$

$$P(z_v^k) = \frac{1}{n}\left(\sum_{i=0}^{k-1}\alpha^i + \alpha^k\left(1+\alpha\left(1+\frac{\left\lceil\frac{\sqrt{n_0\gamma(n_0)}}{\alpha}\right\rceil}{1+\lceil\gamma(n_0)\rceil}\right)\right)\right) \tag{56}$$

and thus both in $\Theta\big(\frac{\alpha^k n_0^{1/2}\gamma(n_0)^{-1/2}}{n}\big)$, and since $\alpha^k = \Theta(f(n_0)\cdot n_0^{1/2}\gamma(n_0)^{1/2})$, both in $\Theta\big(\frac{f(n_0)\cdot n_0}{n}\big) = \Theta(f(n))$. Their difference $P(z_u^k) - P(z_v^k)$ is equal to $\frac{1}{n}\alpha^{k+2}\left\lceil\frac{c\eta}{\alpha^2}\sqrt{\frac{n_0}{\gamma(n_0)}}\right\rceil$, and therefore

$$\frac{P(z_u^k)-P(z_v^k)}{P(z_v^k)} = \frac{\frac{1}{n}\alpha^{k+2}\left\lceil\frac{c\eta}{\alpha^2}\sqrt{\frac{n_0}{\gamma(n_0)}}\right\rceil}{\frac{1}{n}\left(\sum_{i=0}^{k-1}\alpha^i+\alpha^k\left(1+\alpha\left(1+\frac{\left\lceil\frac{\sqrt{n_0\gamma(n_0)}}{\alpha}\right\rceil}{1+\lceil\gamma(n_0)\rceil}\right)\right)\right)} \geq \frac{\alpha^k c\eta\sqrt{\frac{n_0}{\gamma(n_0)}}}{\left(\frac{1}{1-\alpha}+\alpha^k 2\sqrt{\frac{n_0}{\gamma(n_0)}}\right)} \tag{57}$$

and by the choice of k and since $f(n_0) \geq \frac{1}{n_0}$, then $\alpha < \alpha^k\sqrt{\frac{n_0}{\gamma(n_0)}} \leq 1$ and the last term of Equation 57 is greater than $\frac{c\alpha\eta}{1/(1-\alpha)+2\alpha}$, which is greater than or equal to η for any $c \geq 2+\frac{1}{\alpha(1-\alpha)}$.

The family $\mathcal{F}_n$ consists again of all the graphs isomorphic to G, and we give in input to the algorithm, in random order, the ids of z_u^k and z_v^k in a graph drawn uniformly at random from $\mathcal{F}_n$. But returning the correct ranking of z_u^k and z_v^k is equivalent to returning that of u and v, therefore the $\Omega\big(n^{\frac{1}{2}}\gamma(n)^{\frac{1}{2}}\big)$ lower bound obtained for $f(n) = n^{-\frac{1}{2}}\gamma(n)^{-\frac{1}{2}}$ must hold; and for $f(n) < n^{-\frac{1}{2}}\gamma(n)^{-\frac{1}{2}}$, this bound is equivalent to $\Omega\big(\min\big(n^{\frac{1}{2}}\gamma(n)^{\frac{1}{2}}, \frac{1}{f(n)}\big)\big)$.

We thus have a bound of $\Omega\big(\min\big(n^{\frac{1}{2}}\gamma(n)^{\frac{1}{2}}, \frac{1}{f(n)}\big)\big)$ for any $1 \leq \gamma(n) \leq n^{\frac{1}{3}}$ and $\frac{1}{n} \leq f(n) \leq 1$.

Generalization of $f(n)$ and $\gamma(n)$. To remove the assumption that $f(n)$ and $\gamma(n)$ are respectively in $\Theta(f(n_0))$ and $\Theta(\gamma(n_0))$ whenever $n = \Theta(n_0)$, one must build the graphs as a function of n instead of n_0. Crucially, we must then properly rescale the number of nodes at each level in order to ensure that the graph has size exactly n while still guaranteeing that the scores of u and v are not within a factor $(1+\eta)$ of each other. We show how to perform this "rescale" operation only for the case $\gamma(n) > n^{\frac{1}{3}}$ (see above); the case $1 \leq \gamma(n) \leq n^{\frac{1}{3}}$ is completely analogous. For $n^{-\frac{2}{3}} \leq f(n) \leq 1$, consider three arbitrarily small, positive reals $\lambda_0, \lambda_1, \lambda_2$. The graph has the same structure as shown above (see Figure 2), but each subgraph has now $\lceil\lambda_1 n\sqrt{f(n)}\rceil$ level-1 nodes (not counting s_u and s_v) each having $\left\lceil\frac{\lambda_0}{\sqrt{f(n)}}\right\rceil$ additional children besides u or v (except for s_u and s_v), and $\lceil\lambda_2 n f(n)\rceil$ level-2 nodes. The size of the graph is $\Theta(\lambda_0\lambda_1 n + \lambda_2 n f(n)) = O(n(\lambda_0\lambda_1 + \lambda_2))$; we can thus make it exactly equal to n by choosing sufficiently small lambdas and then adding isolated nodes if necessary. Note that now both the scores of the nodes and the lower bound still satisfy the thesis, even if $f(\Theta(n)) \notin \Theta(n)$. We can further choose λ_0 and λ_1 such that also $\frac{\lambda_1}{\lambda_0}$ is sufficiently small; and this ensures that the score contribution of the level-2 nodes is large enough to have the scores not within a factor $(1+\eta)$ of each other, for any n larger than a constant factor depending only on $\eta, \alpha, \lambda_0, \lambda_1, \lambda_2$. Finally, we must ensure that the contribution of the level-2 nodes is sufficient to keep the scores differing by more than $(1+\eta)$ when adapting the graph for $f(n) < n^{-\frac{2}{3}}$, even when $f(n) = \Theta(\frac{1}{n})$ (i.e. when the contribution of the nodes nearest to z_u^k and z_v^k is also in $\Theta(\frac{1}{n})$ and may bring $P(z_u^k)$ and $P(z_v^k)$ too close). To do this, we simply reduce the length of the chain of nodes leading to z_u^k and z_v^k by a sufficiently large constant additive factor (note that this does not change the value of the scores asymptotically), until the score contribution from the level-2 nodes becomes

sufficiently large.

Assembling the lower bound. For any $\gamma(n) > n^{\frac{1}{3}}$ and any $\frac{1}{n} \leq f(n) \leq 1$, we have proven a lower bound of $\Omega\big(\min\big(\frac{1}{f(n)}, n^{2/3}\big)\big) = \Omega\big(\min\big(\frac{1}{f(n)}, n^{2/3}, n^{\frac{1}{2}}\gamma(n)^{\frac{1}{2}}\big)\big)$ queries; and for any $1 \leq \gamma(n) \leq n^{\frac{1}{3}}$ and any $\frac{1}{n} \leq f(n) < 1$, we have also proven a lower bound of $\Omega\big(\min\big(n^{\frac{1}{2}}\gamma(n)^{\frac{1}{2}}, \frac{1}{f(n)}\big)\big) = \Omega\big(\min\big(\frac{1}{f(n)}, n^{2/3}, n^{\frac{1}{2}}\gamma(n)^{\frac{1}{2}}\big)\big)$ queries. Juxtaposing the two gives a general lower bound of $\Omega\big(\min\big(\frac{1}{f(n)}, n^{2/3}, n^{\frac{1}{2}}\gamma(n)^{\frac{1}{2}}\big)\big)$ queries, which concludes the proof.