

# Local Approximation of PageRank and Reverse PageRank\*

Ziv Bar-Yossef  
Department of Electrical Engineering  
Technion, Haifa, Israel  
and  
Google Haifa Engineering Center  
Haifa, Israel  
zivby@ee.technion.ac.il

Li-Tal Mashiach  
Department of Computer Science  
Technion, Haifa, Israel  
litalma@cs.technion.ac.il

## ABSTRACT

We consider the problem of approximating the PageRank of a target node using only local information provided by a link server. We prove that local approximation of PageRank is feasible if and only if the graph has low in-degree and admits fast PageRank convergence. While natural graphs, such as the web graph, are abundant with high in-degree nodes, making local PageRank approximation too costly, we show that *reverse* natural graphs tend to have low in-degree while maintaining fast PageRank convergence. It follows that calculating *Reverse PageRank* locally is frequently more feasible than computing PageRank locally. Finally, we demonstrate the usefulness of Reverse PageRank in five different applications.

**Categories and Subject Descriptors:** H.3.3: Information Search and Retrieval.

**General Terms:** Algorithms.

**Keywords:** PageRank, reverse PageRank, lower bounds, local approximation.

## 1. INTRODUCTION

Over the past decade PageRank [6] has become one of the most popular methods for ranking nodes by their “prominence” in a network.<sup>1</sup> PageRank’s underlying idea is simple but powerful: a prominent node is one that is “supported” (linked to) by other prominent nodes. PageRank was originally introduced as means for ranking web pages in search results. Since then it has found uses in many other domains, such measuring centrality in social networks, evaluating the importance of scientific publications, prioritizing pages in a crawler’s frontier, personalizing search results, combating spam, measuring trust, selecting pages for indexing, and more. While the significance of PageRank in ranking search results seems to have diminished, due to the emergence of other effective alternatives (e.g., clickthrough-based measures), it is still an important tool in search infrastructure, social networks, and analysis of large graphs.

**Local PageRank approximation.** The vast majority of algorithms for computing PageRank, whether they are centralized, parallel, or decentralized, have focused on *global* computation of the PageRank

vector. That is, PageRank scores for *all* the graph’s nodes are computed. While in many applications of PageRank a global computation is needed, there are situations in which one is interested in computing PageRank scores for just a small subset of the nodes.

Consider, for instance, a web site owner (e.g. a small or a large business), who would like to promote the web site in search engine rankings in order to attract traffic of potential clients. As PageRank is used by search engines to determine whether to crawl/index pages and to calculate their relevance scores, tracking the PageRank of the web site would enable the web site owner to better understand its position in search engine rankings and potentially take actions to improve the web site PageRank. In this case, the web site owner is interested only in the PageRank score of his own web site (and maybe also in the scores of his competitors’ web sites), but not in the PageRank scores of all other web pages.

Major search engines choose to keep the PageRank scores of web pages confidential, since there are many variations of the PageRank formula, and making the exact PageRank values public may enable spammers to promote illegitimate web sites. Some search engines publish crude PageRank values (e.g., through the Google Toolbar), but these are usually given in a 1 to 10 logarithmic scale. Users who wish to obtain more accurate PageRank scores for pages of their choice are left to compute them on their own. Global PageRank computation for the entire web graph is out of the question for most users, as it requires significant resources and knowhow. This brings up the following natural question: can one compute the PageRank score of a single web page using reasonable resources?

The same question arises in other natural contexts, where PageRank is used. For example, a Facebook<sup>2</sup> user may be interested in measuring her PageRank popularity by probing the friendship graph. Can this be done efficiently without traversing the whole network?

Chen, Gan, and Suel [1] were the first to introduce the problem of *local PageRank approximation*. Suppose we are given access to a large graph  $G$  through a *link server*, which for every given query node  $x$ , returns the edges incident to  $x$  (both incoming and outgoing).<sup>3</sup> Can we then use a small number of queries to the link server to approximate the PageRank score of a target node  $x$  to within high precision?

Chen *et al.* proposed an algorithm for solving this problem. Their algorithm crawls backwards a small subgraph around the target node, applies various heuristics to guess the PageRank scores of the nodes at the boundary of this subgraph, and then computes the

\*A full version of this paper is available at <http://www.ee.technion.ac.il/people/zivby>. This work was supported by the European Commission Marie Curie International Re-integration Grant and by the Israel Science Foundation.

<sup>1</sup>According to Google Scholar (<http://scholar.google.com>), as of April 2008 the PageRank paper has 1,973 citations.

<sup>2</sup><http://www.facebook.com/>.

<sup>3</sup>If  $G$  is the web graph, out-links can be extracted from the content of  $x$  itself and in-links can be retrieved from search engines using the `link:query`. As opposed to PageRank scores, in-links are information that search engines are willing to disclose.

PageRank of the target node within this subgraph. Chen *et al.* empirically showed this algorithm to provide good approximations on average. However, they noted that high in-degree nodes sometimes make the algorithm either very expensive or inaccurate.

**Lower bounds.** In this work we study the limits of local PageRank approximation. We identify two factors that make local PageRank approximation hard on certain graphs: (1) the existence of high in-degree nodes; (2) slow convergence of the PageRank random walk.<sup>4</sup>

In order to demonstrate the effect of high in-degree nodes, we exhibit for every  $n$  a family of graphs of size  $n$  whose maximum in-degree is high ( $\Omega(\sqrt{n})$ ) and on which any algorithm would need to send  $\Omega(\sqrt{n})$  queries to the link server in order to obtain accurate PageRank approximations. For very large  $n$ , fetching  $\sqrt{n}$  pages from the network or sending  $\sqrt{n}$  queries to a search engine is very costly (for example, for the web graph  $n \geq 10B$ , and thus  $\sqrt{n} \geq 128K$ ). The lower bound we prove applies to both randomized and deterministic algorithms. For deterministic algorithms, we are able to prove an even stronger (and optimal)  $\Omega(n)$  lower bound.

Similarly, to demonstrate the effect of slow PageRank convergence, we present a family of graphs on which the PageRank random walk converges rather slowly (in  $\Omega(\log n)$  steps) and on which every algorithm needs to submit  $\Omega(n^{\frac{1}{2}-\epsilon})$  queries in order to obtain good PageRank approximations ( $\epsilon > 0$  is a small constant that depends on the PageRank damping factor). Again, this lower bound holds for both randomized and deterministic algorithms. For deterministic algorithms, we show an optimal  $\Omega(n)$  lower bound.

We note that the two lower bounds do not subsume each other, as the family of hard graphs constructed in the first lower bound has very fast PageRank convergence (2 iterations), while the family of hard graphs constructed in the second lower bound has bounded in-degree (2).

**Sufficiency.** Having proved that local PageRank approximation is hard for graphs that either have high in-degree or do not admit quick PageRank convergence, it is natural to ask whether local PageRank approximation is feasible for graphs of bounded in-degree and on which PageRank converges quickly. We present a variation of the algorithm by Chen *et al.* and prove that it works well for such graphs: if the PageRank random walk converges on the graph in  $r$  steps, then the algorithm needs to crawl backwards a subgraph of radius  $r$  around the target node. In particular, if the maximum in-degree in this subgraph is  $d$ , then the algorithm requires at most  $d^r$  queries to the link server. This demonstrates that the two conditions we showed to be necessary for fast local PageRank approximation are also sufficient.

**PageRank vs. Reverse PageRank.** As natural graphs, like the web graph and social networks, are abundant with high in-degree nodes, our first lower bound suggests that local PageRank approximation is frequently infeasible to do on such graphs. We substantiate this observation with an empirical analysis of a 280,000 crawl of the [www.stanford.edu](http://www.stanford.edu) site.<sup>5</sup> We show that this graph has a relatively high in-degree (38,606) and as a result the sizes of subgraphs of even small radii around target nodes tend to be high. These findings provide analytical and empirical explanations for the difficulties encountered by Chen *et al.*

We then demonstrate that *reverse* natural graphs (the graphs ob-

tained by reversing the directions of all links) are more suitable for local PageRank approximation. By analyzing the [stanford.edu](http://stanford.edu) crawl, we show that the reverse web graph, like the web graph, admits quick PageRank convergence (on 80% nodes of the reverse graph, PageRank converged within 20 iterations). We also show that the reverse graph has a much lower in-degree (only 255 in the [stanford.edu](http://stanford.edu) crawl), and thus crawl growth rates around target nodes are moderate. These findings hint that local PageRank approximation should be feasible on the reverse graph.

To test this hypothesis, we measured the performance of our algorithm on a sample of nodes from the [stanford.edu](http://stanford.edu) graph. To do the comparison, we selected random nodes from the graph as follows. We ordered all the nodes in the graph by their PageRank, from highest to lowest. We divided the nodes into buckets of exponentially increasing sizes (the first bucket had the top 12 nodes, the second one had the next 24 nodes, and so on). We picked from each bucket 100 random nodes (if the bucket was smaller we took all its nodes). We then calculated, for each bucket, the average cost (number of queries to the link server) of the runs on samples from that bucket. The plot for the reverse graph was constructed analogously.

The results of this experiment show that the cost of the algorithm on the reverse graph is significantly lower than on the regular graph, especially for highly ranked nodes. For example, the average cost of the algorithm on the first bucket of PageRank was three times higher than the cost of the algorithm on the first bucket of *Reverse PageRank* (PageRank of the reverse graph; “RPR” in short). On the other hand, for the low ranked nodes, the performance of the algorithm on the reverse graph and on the regular graph are almost the same. For example, the average cost of the algorithm on the last bucket of PageRank was 13 and for Reverse PageRank it was 14.

We conclude from the above that computing Reverse PageRank is more feasible to do locally than computing regular PageRank. Social networks and other natural graphs possess similar properties to the web graph (power law degrees, high in-degree vs. low out-degree) and are thus expected to exhibit similar behavior.

**Applications of Reverse PageRank.** While locally approximating RPR is easier than locally approximating PageRank, why would one want to compute RPR in the first place? We observe that RPR has a multitude of applications. It has been used before to select good seeds for the TrustRank measure [3], to detect highly influential nodes in social networks [5], and to find hubs in the web graph [2]. We present two additional novel applications of RPR: (1) finding good seeds for crawling; and (2) measuring the “semantic relatedness” of concepts in a taxonomy.

## 2. REFERENCES

- [1] Y. Chen, Q. Gan, and T. Suel. Local methods for estimating PageRank values. In *CIKM*, pages 381–389, 2004.
- [2] D. Fogaras. Where to start browsing the Web? In *IICS*, pages 65–79, 2003.
- [3] Z. Gyöngyi, H. Garcia-Molina, and J. Pedersen. Combating Web Spam with TrustRank. In *VLDB*, pages 576–587, 2004.
- [4] T. H. Haveliwala and S. D. Kamvar. The second eigenvalue of the Google matrix. Technical report, Stanford University, 2003.
- [5] A. Java, P. Kolari, T. Finin, and T. Oates. Modeling the spread of influence on the Blogosphere. Technical report, University of Maryland, Baltimore County, 2006.
- [6] L. Page, S. Brin, R. Motwani, and T. Winograd. The PageRank citation ranking: Bringing order to the Web. Technical report, Stanford Digital Library Technologies Project, 1998.

<sup>4</sup>Note that the convergence rate of PageRank on a given graph is an intrinsic property of the graph, not of the particular algorithm used to compute PageRank. The convergence rate is governed by the difference between the first and the second eigenvalues of PageRank’s transition matrix.

<sup>5</sup>While this crawl is relatively small, its structural properties have been shown before to be representative of the whole web graph [4].