Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence

# Fast and Exact Top-k Algorithm for PageRank

**Yasuhiro Fujiwara†, Makoto Nakatsuji‡, Hiroaki Shiokawa†,**
**Takeshi Mishima†, Makoto Onizuka†**
†NTT Software Innovation Center, 3-9-11 Midori-cho Musashino-shi, Tokyo, Japan
‡NTT Service Evolution Laboratories, 1-1 Hikarinooka Yokosuka-shi, Kanagawa, Japan
{fujiwara.yasuhiro, nakatsuji.makoto, shiokawa.hiroaki, mishima.takeshi, onizuka.makoto}@lab.ntt.co.jp

## Abstract

In AI and Web communities, many applications utilize *PageRank*. To obtain high PageRank score nodes, the original approach iteratively computes the PageRank score of each node until convergence by using the whole graph. If the graph is large, this approach is infeasible due to its high computational cost. The goal of this study is to find top-k PageRank score nodes efficiently for a given graph without sacrificing accuracy. Our solution, *F-Rank*, is based on two ideas: (1) It iteratively estimates lower/upper bounds of PageRank scores, and (2) It constructs subgraphs in each iteration by pruning unnecessary nodes and edges to identify top-k nodes. Our theoretical analysis shows that F-Rank guarantees result exactness. Experiments show that F-Rank finds top-k nodes much faster than the original approach.

## Introduction

Graphs are widely used to represent data entities as well as the relationships among them (Nakatsuji et al. 2012; Shiokawa, Fujiwara, and Onizuka 2013; Fujiwara et al. 2013). *PageRank* is the most popular approach for computing the importance of nodes in a graph (Page et al. 1999). It is based on the "*random surfer model*" (Langville and Meyer 2012); web users get bored after several clicks and switch to a random page. Informally, PageRank scores correspond to the stationary distribution of random walks. At each step of PageRank, it randomly selects an outgoing edge from the current node, and, with a certain probability, it randomly jumps to a node in the graph. Due to the effectiveness and solid theoretical foundation of PageRank, it is used by many applications in AI and Web communities recently even though PageRank was originally proposed to enhance the effectiveness of information retrieval:

**Comment summarization** Online commenting enables news sites to extend their user volume as well as to increase advertising revenue (Shmueli et al. 2012). However, it is time-consuming to read all comments of an article since a popular news article may easily accumulate thousands of comments within a short time period. The PageRank-based approach suggested by Khabiri et al. can select the best top-k comments for summarization (Khabiri, Caverlee, and Hsu 2011). They obtain a graph in an ad-hoc manner; they generate nodes from comments, and a node has an edge to another node if the two nodes (comments) share terms. Comment importance of a news article is computed by PageRank. Therefore, they search top-k nodes and identify the obtained nodes as the summary comments of the news article. Their approach yields more accurate summaries than the LexRank-based approach (Erkan and Radev 2004).

**Synonym expansion** Synonym expansion is the task of replacing a target word in a given sentence with another, more suitable, word. This is useful for question answering (Dagan et al. 2006) or text simplification (McCarthy 2002). Sinha et al. applied PageRank to synonym expansion (Sinha and Mihalcea 2011). For a given sentence and target word, they collect all the synonyms from WordNet (Fellbaum and Miller 1998) on the fly. The entire set of candidate synonyms, along with the words in the sentence, are used to generate nodes. Edges are generated according to word similarities. They compute PageRank scores of candidate nodes (synonyms). And they select top-k nodes as relevant synonyms. Their approach is superior to the contextual fitness approach (Sinha and Mihalcea 2009) in finding effective synonyms.

**Web content extraction** The mobile phone market has been increased recently at a rate of nearly 50%; there are three billion mobile subscribers worldwide. Mobile access is a key to individual productivity by providing opportunities to connect to the web at any time from anywhere (Lee, Yeung, and Yu 2012). However, mobile phones are not ideal platforms for surfing the web since the screens are very small. The approach proposed by Yin et al. utilizes PageRank to convert web pages to those suitable for mobile phones (Yin and Lee 2004). They divide the web page into inseparable elements such as images or text paragraphs to obtain nodes. Edges are generated from word similarities and physical proximities of the elements within the same page. They obtain the most important topic of a web page by detecting the top element in PageRank. The web page is reformatted by the detected element and its relevant elements. Their approach provides more effective results than cosine similarity (Manning and Schuetze 1999) based approach.

Although PageRank is effective, it is difficult for the original approach to realize quick response for large graphs since

PageRank scores are computed from the whole graph until convergence. This problem has led to the introduction of acceleration approaches. Surveys of recent approaches are reported in (Bahmani, Chowdhury, and Goel 2010) and (Fujiwara et al. 2012b). However, the approaches have technical limitations. Broadly speaking, there are three approaches of linear algebraic, score update, and Monte Carlo.

A linear algebraic scheme was proposed by Kamvar et al. (Kamvar, Haveliwala, and Golub 2004). The linear algebraic approach halts score computation of a node once the PageRank score of the node converges. However, there is a serious theoretical problem that this approach may not converge and results may not be correct. Gleich et al. studied another linear algebraic scheme; they applied Krylov subspace methods (Watkins 2007) for PageRank instead of the iterative power method which is used in the original approach (Gleich, Zhukov, and Berkhin 2004). The approach is faster than the original approach. However, the convergence of the linear algebraic approach is not stationary, i.e., PageRank scores yielded by the approach behave erratically.

Score update is another approach; the idea is that an incremental change in the graph mostly impacts on only local PageRank scores. Based on this idea, several approximate schemes have been investigated (Chien et al. 2003; McSherry 2005; Langville and Meyer 2006; Bahmani et al. 2012). However, graphs do not always change in an incremental manner; graphs can be fully known only after a query is submitted to the above applications. For example, in synonym expansion, graphs are generated in an ad-hoc manner from the given sentence and target word. Furthermore, it is difficult for these approximate approaches to enhance the quality of real applications (Fujiwara et al. 2012a).

The Monte Carlo approach, studied by Bahmani et al., precomputes random walks and stores them in a disk (Bahmani, Chowdhury, and Goel 2010). To compute approximate score of a node, it counts the total number of times the node is visited by the random walks. Unfortunately, their approach has high I/O cost since it is disk-based. Avrachenkov et al. proposed several Monte Carlo schemes (Avrachenkov et al. 2007). The Monte Carlo approaches can approximately compute the top-k nodes in ad-hoc style since they perform random walks on a given graph on the fly. However, the Monte Carlo approaches require that the number of random walks to be set, which induces a trade-off between efficiency and approximation quality.

To overcome the limitations of the previous approaches, the research problem in this paper is posed as follows:

**Problem (Top-k search for PageRank).**

**Given:** *Arbitrary graph $\mathbb{G}$ and required number of answer nodes $k$.*

**Find:** *Top-k nodes with respect to their PageRank scores exactly and efficiently.*

We propose a novel approach, *F-Rank*, that finds top-k nodes efficiently and exactly for PageRank. In order to reduce computation cost, (1) we estimate lower/upper bounds of PageRank scores in each iteration, and (2) we dynamically construct subgraphs to identify top-k nodes efficiently. F-Rank has the following attractive characteristics:

Table 1: Definition of main symbols.

| Symbol | Definition |
|---|---|
| $N$ | Number of nodes in the graph |
| $M$ | Number of edges in the graph |
| $T$ | Number of iterations by the original approach |
| $k$ | Number of required answer nodes for top-k search |
| $\epsilon_i$ | $k$-th highestlower estimation in the $i$-th iteration |
| $\mathbb{G}$ | Given graph |
| $\mathbb{V}$ | Set of nodes in $\mathbb{G}$ |
| $\mathbb{E}$ | Set of edges in $\mathbb{G}$ |
| $\mathbb{C}$ | Set of candidate nodes |
| $\mathbb{R}$ | Set of reachable nodes to candidate nodes |
| $\mathbf{W}$ | $N \times N$ column normalized adjacent matrix of $\mathbb{G}$ |
| $\mathbf{p}$ | $N \times 1$ PageRank vector |
| $\overline{\mathbf{p}}$ | $N \times 1$ upper bounding PageRank vector |
| $\underline{\mathbf{p}}$ | $N \times 1$ lower bounding PageRank vector |

- **Fast**: By utilizing the above ideas, F-Rank is much faster than existing approaches.
- **Exact**: F-Rank does not sacrifice accuracy; it returns the exact top-k nodes.
- **Flexible**: F-Rank does not require any precomputation step; it can effectively handle ad-hoc search for any arbitrary graphs and number of answer nodes.
- **Parameter-free**: Our approach does not require any inner-parameters. Thus it provides the user with a simple solution to PageRank-based applications.

By providing the exact solution in a highly efficient manner, F-Rank will allow many more PageRank-based applications to be developed in the future.

## Preliminary

We formally define the notations and introduce the background of this paper. Table 1 lists the main symbols and their definitions. Starting from a random node, PageRank performs random walks by iteratively following an edge to another node at each step with probability $s(0 < s < 1)$. Additionally, at each step, it jumps to a random node with probability $1 - s$. Let $\mathbb{G} = \{\mathbb{V}, \mathbb{E}\}$ be a given graph where $\mathbb{V}$ and $\mathbb{E}$ are set of nodes and edges, respectively. $\mathbf{p}$ represents a column vector whose $u$-th element $p[u]$ denotes the PageRank score of node $u$. $\mathbf{e}$ is a column vector where every element is set to $1/N$; $N$ is the number of nodes in the graph. Also let $\mathbf{W}$ be the column normalized adjacency matrix where its element $W[u, v]$ gives the probability of node $v$ being the next state given that the current state is node $u$. The PageRank score of nodes can be obtained by recursively applying the following equation until convergence:

$$\mathbf{p}_i = s\mathbf{W}\mathbf{p}_{i-1} + (1 - s)\mathbf{e} \quad (1)$$

where $\mathbf{p}_i$ is set to $\mathbf{e}$ if $i = 0$. The convergence of the equation is guaranteed (Langville and Meyer 2012). This approach updates the PageRank score of each node that is obtained in the previous iteration by exploiting the whole graph. It needs $O((N + M)T)$ time where $M$ is the number of edges in the graph and $T$ is the number of iteration steps until convergence. This incurs high computational cost for large graphs.

## Proposed method

We present our proposal, F-Rank, that exactly and efficiently finds top-k nodes for PageRank. First, we give an overview of the ideas underlying F-Rank. That is followed by a full description including a top-k search algorithm.

### Ideas

We introduce the idea of estimating the lower and upper bounds of PageRank scores to reduce the computational cost. Instead of using the whole graph to compute PageRank scores, we obtain subgraphs to compute the estimations by pruning unnecessary nodes and edges in the iterations.

This idea has several advantages. First, we can terminate the iterations without waiting for convergence if $k$ nodes are identified as the candidate nodes of the top-k search. This means that our approach needs fewer iterations than the original approach. Second, we can exactly detect the top-k nodes even though we estimate the lower/upper bounds. This is because we can safely discard unpromising nodes by the estimations. Third, we can efficiently construct the subgraphs on the fly for any given graph by utilizing the estimations. The estimations allow us to detect the nodes and edges that do not need to be utilized to find the top-k nodes. Based on this idea, we dynamically construct subgraphs that have only the necessary nodes and edges (Fujiwara, Irie, and Kitahara 2011), whereas the original approach must use the whole graph to compute PageRank scores. Finally, the estimations, as well as the subgraphs, do not require any user-defined inner-parameter, whereas previous approaches need careful inner-parameter setting. For example, Monte Carlo schemes require setting of the number of random walks, which induces a trade-off between computation time and approximation accuracy (Avrachenkov et al. 2007; Bahmani, Chowdhury, and Goel 2010). Thus our approach, F-Rank, is user-friendly.

### Lower and upper estimations

We compute lower/upper estimations to obtain subgraphs. We introduce the lower/upper bound estimations and describe their properties. In the $i$-th iteration ($i = 0, 1, 2, \ldots$), we compute the estimations for the candidate node set $\mathbb{C}_i$. The next section describes our approach to obtaining the candidate node set $\mathbb{C}_i$. To compute the upper estimation, we use $\mathbb{R}_i$, the set of reachable nodes to any node in $\mathbb{C}_i$. Node $u$ is reachable to node $v$ if there is a path from node $u$ to $v$ (Bryce and Kambhampati 2007). We also use an $N \times 1$ vector of the maximum edge weights, $\overline{\mathbf{W}}$, where each element is given by $\overline{W}[u] = \max\{W[u, v] : v \in \mathbb{V}\}$. Let $\mathbf{r}_i$ be the $N \times 1$ probability vector of $i$ length random walk; $\mathbf{r}_i$ is computed as $\mathbf{r}_i = \mathbf{W}^i\mathbf{e}$ by using the $i$-th power of the adjacent matrix $\mathbf{W}$. If $i = 0$, $\mathbf{W}^i = \mathbf{I}$ where $\mathbf{I}$ is the identity matrix.

We define the lower and upper bounding PageRank vectors in the $i$-th iteration as $\underline{\mathbf{p}}_i$ and $\overline{\mathbf{p}}_i$, respectively:

**Definition 1 (Lower estimation)** *The lower estimation in the $i$-th iteration is given as follows:*

$$\underline{\mathbf{p}}_i = (1 - s)\sum_{j=0}^{i} s^j \mathbf{r}_j \quad (2)$$

**Definition 2 (Upper estimation)** *The following equation gives the upper estimation in the $i$-th iteration:*

$$\overline{\mathbf{p}}_i = (1 - s)\sum_{j=0}^{i} s^j \mathbf{r}_j + s^{i+1}\mathbf{r}_i + \Delta_i \sigma_i \overline{\mathbf{W}} \quad (3)$$

*In Equation (3), $\sigma_i = s^{i+1}(1 - s)^{-1}$ and $\Delta_i$ is computed from the elements in $\mathbf{r}^i$ as follows:*

$$\Delta_i = \begin{cases} 1 & (i = 0) \\ \sum_{u \in \mathbb{R}_i} \Delta_i[u] & (i \neq 0) \end{cases} \quad (4)$$

*where $\Delta_i[u] = \max\{r_i[u] - r_{i-1}[u], 0\}$.*

We recursively compute the lower and upper estimations in each iteration by using the random walk probabilities as shown in Definition 1 and 2. The next section introduces our incremental approach to computing the estimations from subgraphs. The theoretical aspects are as follows:

**Lemma 1 (Lower estimation)** *For $u$-th elements in $\mathbf{p}$ and $\underline{\mathbf{p}}_i$, $\underline{p}_i[u] \leq p[u]$ holds in the $i$-th iteration.*

**Proof** From Equation (1), we have

$$\begin{aligned}\mathbf{p}_i &= s\mathbf{W}\mathbf{p}_{i-1} + (1 - s)\mathbf{e} = s^2\mathbf{W}^2\mathbf{p}_{i-2} + (1 - s)(s\mathbf{W}\mathbf{e} + \mathbf{e}) \\ &= s^i\mathbf{W}^i\mathbf{p}_0 + (1 - s)(s^{i-1}\mathbf{W}^{i-1}\mathbf{e} + s^{i-2}\mathbf{W}^{i-2}\mathbf{e} + \ldots + \mathbf{e}) \\ &= s^i\mathbf{W}^i\mathbf{e} + (1 - s)\textstyle\sum_{j=0}^{i-1}(s^j\mathbf{W}^j\mathbf{e})\end{aligned}$$

Because the PageRank score of each node is the convergence value in Equation (1), we have $\mathbf{p} = \mathbf{p}_\infty$. Therefore,

$$\mathbf{p} = s^\infty\mathbf{W}^\infty\mathbf{e} + (1 - s)\textstyle\sum_{j=0}^{\infty}(s^j\mathbf{W}^j\mathbf{e}) = (1 - s)\sum_{j=0}^{\infty} s^j\mathbf{r}_j$$

since $0 < s < 1$ and the elements in $\mathbf{W}^\infty$ cannot be larger than 1 or smaller than 0. The above equation indicates that the following inequality holds for node $u$:

$$p[u] = (1 - s)\textstyle\sum_{j=0}^{\infty} s^j r_j[u] \geq (1 - s)\sum_{j=0}^{i} s^j r_j[u] = \underline{p}_i[u]$$

which completes the proof. □

**Lemma 2 (Upper estimation)** $\overline{p}_i[u] \geq p[u]$ *holds for $\mathbf{p}$ and $\overline{\mathbf{p}}_i$ in the $i$-th iteration.*

**Proof** As shown in the above proof,

$$\begin{aligned}p[u] =& (1 - s)\textstyle\sum_{j=0}^{\infty} s^j r_j[u] \\ =& (1 - s)\textstyle\sum_{j=0}^{i} s^j r_j[u] + (1 - s)\sum_{j=1}^{\infty} s^{i+j} r_{i+j}[u]\end{aligned}$$

We first show the property that $r_{i+j}[u] \leq r_i[u] + j\Delta_i\overline{W}[u]$. Let $\mathbb{H}_j[u]$ be set of nodes that are reachable to node $u$ by $j$ hops. Note that $\mathbb{H}_j[u] \subseteq \mathbb{R}_i \subseteq \mathbb{V}$. Since $\mathbf{r}_{i+j} - \mathbf{r}_{i+j-1} = \mathbf{W}^{i+j}\mathbf{e} - \mathbf{W}^{i+j-1}\mathbf{e} = \mathbf{W}\mathbf{W}^{j-1}(\mathbf{r}_i - \mathbf{r}_{i-1})$, we have

$$\begin{aligned}& r_{i+j}[u] - r_{i+j-1}[u] \\ =& \textstyle\sum_{v \in \mathbb{H}_1[u]} \sum_{w \in \mathbb{H}_{j-1}[v]} W[u, v]W^{j-1}[v, w](r_i[w] - r_{i-1}[w]) \\ \leq& \textstyle\sum_{w \in \mathbb{H}_{j-1}[v]} \sum_{v \in \mathbb{H}_1[u]} \overline{W}[u]W^{j-1}[v, w]\Delta_i[w] \\ \leq& \overline{W}[u] \textstyle\sum_{w \in \mathbb{R}_i} \Delta_i[w] \left(\sum_{v \in \mathbb{H}_1[u]} W^{j-1}[v, w]\right)\end{aligned}$$

We have $\sum_{v \in \mathbb{H}_1[u]} W^{j-1}[v, w] \leq 1$ since $\mathbf{W}^{j-1}$ is a column normalized matrix. Therefore,

$$r_{i+j}[u] - r_{i+j-1}[u] \leq \overline{W}[u] \textstyle\sum_{w \in \mathbb{R}_i} \Delta_i[w] = \Delta_i\overline{W}[u]$$

As a result,

$$r_{i+j}[u] \leq r_{i+j-1}[u] + \Delta_i\overline{W}[u] \leq \ldots \leq r_i[u] + j\Delta_i\overline{W}[u]$$

By exploiting this property, we have

$$(1-s)\sum_{j=1}^{\infty} s^{i+j} r_{i+j}[u] \leq (1-s)\sum_{j=1}^{\infty}(s^{i+j} r_i[u] + j s^{i+j} \Delta_i \overline{W}[u])$$

Since $\sum_{j=1}^{\infty} s^{i+j} \leq \frac{s^{i+1}}{1-s}$ and $\sum_{j=1}^{\infty} j s^{i+j} \leq \frac{\sigma[i]}{1-s}$,

$$(1-s)\sum_{j=1}^{\infty} s^{i+j} r_{i+j}[u] \leq s^{i+1} r_i[u] + \Delta_i \sigma[i] \overline{W}[u]$$

Therefore, from Equation (3),

$$p[u] \leq (1-s)\sum_{j=0}^{i} s^j r_j[u] + s^{i+1} r_i[u] + \Delta_i \sigma[i] \overline{W}[u] = \overline{p}_i[u]$$

which completes the proof. □

**Lemma 3 (Convergence of the estimations)** *The estimations converge with PageRank scores;* $\underline{p}_{\infty}[u] = \overline{p}_{\infty}[u] = p[u]$.

**Proof** Omitted due to the space limitation. □
Lemma 3 ensures the convergence of our approach.

## Subgraph

We iteratively compute candidate nodes to find the top-k nodes, and terminate the iterations if the number of candidate nodes is $k$. We compute the estimations *only* for candidate nodes in subgraphs, which are dynamically obtained in each iteration. We show definitions of candidate nodes and subgraphs along with the theoretical properties.

A set of candidate nodes in the $i$-th iteration, $\mathbb{C}_i$, is defined as follows by letting $\epsilon_{i-1}$ be the $k$-th highest lower estimation in the previous $i$–1-th iteration:

**Definition 3 (Candidate nodes)** *The following equation gives the set of candidate nodes in the $i$-th iteration:*

$$\mathbb{C}_i = \begin{cases} \mathbb{V} & (i=0) \\ \{u \in \mathbb{V} : \overline{p}_{i-1}[u] \geq \epsilon_{i-1}\} & (i \neq 0) \end{cases} \quad (5)$$

The above definition shows that a node is a candidate node if its upper estimation is not lower than $\epsilon_{i-1}$. This is because, if the upper estimation of the node is lower than $\epsilon_{i-1}$, the exact PageRank score of the node must be lower than $\epsilon_{i-1}$. We show the theoretical property of node set $\mathbb{C}_i$ as follows:

**Lemma 4 (Candidate nodes)** *If node $u$ is not included in $\mathbb{C}_i$, i.e., $u \notin \mathbb{C}_i$, node $u$ cannot be an answer node.*

**Proof** Let $\epsilon$ be the $k$-th highest exact PageRank score, it is clear that $\epsilon_{i-1} \leq \epsilon$ from Lemma 1. And $\overline{p}_{i-1}[u] \geq p[u]$ from Lemma 2. Let $\mathbb{A}$ be answer nodes, if $i \neq 0$, we have

$$\mathbb{A} = \{u \in \mathbb{V} : p[u] \geq \epsilon\} \subseteq \{u \in \mathbb{V} : \overline{p}_{i-1}[u] \geq \epsilon_{i-1}\} = \mathbb{C}_i$$

If $i = 0$, $\mathbb{A} \subseteq \mathbb{V} = \mathbb{C}_i$, so there is no node such that $u \notin \mathbb{C}_i$. Therefore, if $u \notin \mathbb{C}_i$, node $u$ cannot be an answer node. □
Since $\mathbb{A} \subseteq \mathbb{C}_i$ from Lemma 4, we incrementally compute $\mathbb{C}_i$ from $\mathbb{C}_{i-1}$ in each iteration as follows:

**Definition 4 (Update of candidate nodes)** *If $i \neq 0$, we incrementally compute $\mathbb{C}_i$ in each iteration as follows:*

$$\mathbb{C}_i = \{u \in \mathbb{C}_{i-1} : \overline{p}_{i-1}[u] \geq \epsilon_{i-1}\} \quad (6)$$

**Lemma 5 (Update of candidate nodes)** *In the iteration, the candidate nodes monotonically decrease; $\mathbb{C}_i \subseteq \mathbb{C}_{i-1}$.*

**Proof** Since $\mathbb{C}_i$ is obtained as a subset of $\mathbb{C}_{i-1}$ in Equation (6), it is clear that $\mathbb{C}_i \subseteq \mathbb{C}_{i-1}$. □

We compute the estimations for $\mathbb{C}_i$ by subgraphs. The subgraph in the $i$-th iteration is defined as follows:

**Definition 5 (Subgraph)** *Let $\mathbb{G}_i = \{\mathbb{V}_i, \mathbb{E}_i\}$ be the subgraph in the $i$-th iteration. If $i = 0$, $\mathbb{V}_0$ and $\mathbb{E}_0$ are defined as $\mathbb{V}$ and $\mathbb{E}$, respectively. If $i \neq 0$, $\mathbb{V}_i$ and $\mathbb{E}_i$ are defined as $\mathbb{V}_i = \mathbb{R}_i$ and $\mathbb{E}_i = \{(u,v) \in \mathbb{E} : u \in \mathbb{R}_i, v \in \mathbb{R}_i\}$, respectively, where $(u,v)$ is an edge from node $u$ to $v$.*

We introduce the following lemma for the subgraph.

**Lemma 6 (Subgraph)** *The estimations of the $i$-th iteration for candidate nodes can be obtained from subgraph $\mathbb{G}_i$.*

**Proof** If $i = 0$, this lemma obviously holds since $\mathbb{G}_i$ corresponds to the given graph $\mathbb{G}$. Otherwise, as shown in Definition 1 and 2, the estimations of node $u$ can be computed if the random walk probability of node $u$ is obtained. If node $v$ is *not* reachable to node $u$, the random walk probability of node $v$ does not affect the random walk probability of node $u$. Therefore, node set $\mathbb{R}_i$ and the set of edges which are incident to $\mathbb{R}_i$ need to be processed to obtain the estimations for candidate nodes. □

**Lemma 7 (Monotonic decrease of $\mathbb{G}_i$)** *In the iterations, the subgraphs have the property that $\mathbb{G}_i \subseteq \mathbb{G}_{i-1}$.*

**Proof** Since (1) $\mathbb{R}_i$ is the set of reachable nodes to any node in $\mathbb{C}_i$, and (2) $\mathbb{C}_i \subseteq \mathbb{C}_{i-1}$ from Lemma 5, it is clear that $\mathbb{R}_i \subseteq \mathbb{R}_{i-1}$. Therefore, $\mathbb{G}_i \subseteq \mathbb{G}_{i-1}$ from Definition 5. □
The next section describes our algorithm that constructs subgraphs based on Lemma 7.

We incrementally compute the estimations from subgraphs in the iterations as follows:

**Definition 6 (Incremental estimations)** *The lower and upper estimations are incrementally computed as follows:*

$$\underline{p}_i[u] = \begin{cases} (1-s)/N & (i=0) \\ \underline{p}_{i-1}[u] + (1-s)s^i r_i[u] & (i \neq 0) \end{cases} \quad (7)$$

$$\overline{p}_i[u] = \begin{cases} 1/N + s(1-s)^{-1}\overline{W}[u] & (i=0) \\ \underline{p}_{i-1}[u] + s^i r_i[u] + \Delta_i \sigma_i \overline{W}[u] & (i \neq 0) \end{cases} \quad (8)$$

*where $r_i[u]$ is computed as $r_i[u] = \sum_{v \in \mathbb{V}_i} W[u,v] r_{i-1}[v]$ from subgraph $\mathbb{G}_i$ if $i \neq 0$, and $\mathbf{r}_0 = \mathbf{e}$.*

This definition indicates that (1) if $i = 1$, the estimations of a node are obtained by the probability $s$, the number of nodes in the graph, and the edge weights, and (2) otherwise, we can incrementally update the lower/upper estimations from the lower estimation of the previous iteration. From Definition 6, we have the following property:

**Lemma 8 (Incremental estimations)** *For $u \in \mathbb{C}_i$, the estimations are exactly computed at the cost of $O(1)$ by Definition 6 if the random walk probability is obtained for $v \in \mathbb{V}_i$.*

**Proof** Omitted due to the space limitation. □
This lemma ensures that we can efficiently obtain the estimations for the candidate nodes in the iterations.

## Search algorithm

Algorithm 1, F-Rank, finds the top-k nodes exactly. If $i = 0$, it initializes $\mathbb{C}_0 = \mathbb{V}$ and $\mathbb{G}_0 = \mathbb{G}$ based on Definition 3 and 5, respectively (lines 2-3). Otherwise, it computes $\mathbb{R}_i$ to $\mathbb{C}_i$ in $\mathbb{G}_{i-1}$ by applying breadth-first search inversely (line 7). This is because $\mathbb{G}_i$ has the property that $\mathbb{G}_i \subseteq \mathbb{G}_{i-1}$ from Lemma 7. It constructs $\mathbb{G}_i$ from $\mathbb{R}_i$ from Definition 5 (line 8). It then computes the random walk probability of each node in $\mathbb{G}_i$ (lines 10-12), since the probability of each node is needed to compute the estimations for candidate nodes from Lemma 6. It computes the estimations for $\mathbb{C}_i$ (lines 13-15), and computes $\epsilon_i$ from $\mathbb{C}_i$ (line 16). It updates the candidate node set to $\mathbb{C}_{i+1}$ (line 17). If the size of $\mathbb{C}_{i+1}$ is $k$

**Algorithm 1** F-Rank

**Input:** $\mathbb{G}$, original graph; $k$, number of answer nodes
**Output:** set of top-k nodes

1: $i := 0$;
2: $\mathbb{C}_0 := \mathbb{V}$;
3: $\mathbb{G}_0 := \mathbb{G}$;
4: **repeat**
5: **if** $i \neq 0$ **then**
6: $i := i + 1$;
7: compute $\mathbb{R}_i$ to $\mathbb{C}_i$ in $\mathbb{G}_{i-1}$ by breadth-first search;
8: compute $\mathbb{G}_i$ from $\mathbb{R}_i$;
9: **end if**
10: **for** each node $u \in \mathbb{V}_i$ **do**
11: compute $r_i[u]$ from $\mathbb{G}_i$;
12: **end for**
13: **for** each node $u \in \mathbb{C}_i$ **do**
14: compute $\underline{p}_i[u]$ and $\overline{p}_i[u]$ by Equation (7) and Equation (8), respectively;
15: **end for**
16: compute $\epsilon_i$ from $\mathbb{C}_i$;
17: compute $\mathbb{C}_{i+1}$ from $\epsilon_i$ and $\mathbb{C}_i$ by Equation (6);
18: **until** $|\mathbb{C}_{i+1}| = k$
19: **return** $\mathbb{C}_{i+1}$;

(i.e., $|\mathbb{C}_{i+1}| = k$), all nodes in $\mathbb{C}_{i+1}$ must be answer nodes from Lemma 4. Therefore, it terminates the iteration process (line 18) and returns the answer nodes (line 19).

As shown in Algorithm 1, F-Rank does not need precomputation and does not require any inner-parameters; F-Rank is a simple solution to finding the top-k nodes for PageRank.

We provide theoretical analyses that address the search results and the computational cost of F-Rank. The following theorem shows that F-Rank finds the top-k nodes exactly:

**Theorem 1 (Exactness in top-k search)** *F-Rank finds the top-k nodes exactly with respect to PageRank scores.*

**Proof** In the $i$-th iteration, F-Rank prunes node $u$ if $\overline{p}_i[u] < \epsilon_i$. Since $\epsilon_i \leq \epsilon$ from Lemma 1 and $\overline{p}_i[u] \geq p_i[u]$ from Lemma 2, the answer nodes cannot be pruned by F-Rank. If node $u$ is not an answer node, $\overline{p}_i[u] < \epsilon$ holds in at least one iteration from Lemma 2; the node must be pruned with its upper estimation in that iteration. Therefore, the result of F-Rank is equivalent to that of the original approach. □

We discuss the computational cost of F-Rank. Let $n$ and $m$ be the average number of nodes and edges of the subgraphs, respectively. $c$ and $t$ be the average number of candidate nodes and the number of iterations in F-Rank, respectively. It is clear that $c \leq n$. Note that, the original approach requires $O((N + M)T)$ time.

**Theorem 2 (Computational cost)** *F-Rank needs* $O((n + m + \log c \log k)t)$ *time to obtain the top-k nodes.*

**Proof** F-Rank first constructs the subgraphs by breadth-first search in $O((n + m)t)$ time. It computes the random walk probabilities for each node in the subgraphs, which needs $O((n + m)t)$ time. The estimations of the candidate nodes are obtained at $O(ct)$ since it needs $O(1)$ time to compute the lower/upper estimations of a node in each iteration (Lemma 8). It requires $O(\log c \log k)$ time to compute $\epsilon_i$ from the candidate nodes by the lower estimations in each iteration. This is because (1) the $k$-th highest lower estimation can be updated at $O(\log k)$ time by using Fibonacci heaps (Mehlhorn and Sanders 2010) if a new $k$-th node is detected from the candidate nodes, and (2) the expected number of update is $O(\log c)$ by randomly accessing the candidate nodes (Cormen et al. 2009). $\mathbb{C}_{i+1}$ is obtained at $O(ct)$ time from $\mathbb{C}_i$ by using $\epsilon_i$ and the lower estimations. Thus, F-Rank requires $O((n + m + \log c \log k)t)$ time. □

# Experimental evaluation

We performed experiments to confirm the effectiveness of F-Rank. Our experiments were designed to show that:

Table 2: Score of each inner-parameter.

| Parameter | Dataset | | |
|---|---|---|---|
| | P2P | Web | Wikipedia |
| $N$ | $6.26 \times 10^4$ | $3.26 \times 10^5$ | $2.39 \times 10^6$ |
| $c$ | $3.16 \times 10^4$ | $1.49 \times 10^5$ | $4.00 \times 10^5$ |
| $n$ | $4.69 \times 10^4$ | $2.70 \times 10^5$ | $6.29 \times 10^5$ |
| $M$ | $1.48 \times 10^5$ | $3.22 \times 10^6$ | $5.02 \times 10^6$ |
| $m$ | $1.20 \times 10^5$ | $3.06 \times 10^6$ | $2.44 \times 10^6$ |
| $T$ | 18 | 116 | 97 |
| $t$ | 9 | 33 | 21 |

- **Efficiency**: F-Rank outperforms the original approach (Page et al. 1999) in terms of computation time.
- **Exactness**: F-Rank finds the top-k nodes exactly unlike the Monte Carlo approach (Avrachenkov et al. 2007).

The experiments use the following three public datasets:

- **P2P**[1]: This graph is a snapshot of the Gnutella peer-to-peer file sharing network. In this graph, nodes represent hosts in the Gnutella network topology and edges represent connections between the hosts. The number of nodes and edges are $62,586$ and $147,892$, respectively.
- **Web**[2]: CNR (Italy's National Research Council) is a public research organization. This graph is the result of a crawl of the Italian CNR domain where nodes and edges correspond to pages and hyperlinks, respectively. This graph has $325,557$ nodes and $3,216,152$ edges.
- **Wikipedia**[3]: Each registered Wikipedia user has a talk page, that other users can edit for discussion. In this graph, nodes are Wikipedia users. An edge from node $u$ to $v$ represents that user $u$ edited a talk page of user $v$. There are $2,394,385$ nodes and $5,021,410$ edges.

Note that graph sizes increase in the order of P2P, Web, and Wikipedia. We set $s = 0.85$, the same as the original paper (Page et al. 1999). All experiments were conducted on a Linux 3.33 GHz Intel Xeon server with 32GB of main memory. We implemented all approaches using GCC.

## Efficiency

We evaluated the search time of F-Rank and the original approach. Figure 1 shows the results. In Figure 1, the results of F-Rank are indicated by "F-Rank($k$)" where $k$ is the number of answer nodes. In the original approach, the the iterations are terminated when the residual 1-norm dropped below $10^{-10}$, as is used by a previous study (Langville and Meyer 2006) [4]. Note that, for the original approach, the number of answer nodes does not have an impact on the search time since it computes PageRank scores of all nodes. Table 2 details the inner-parameters in each approach where $k = 50$. Note that these parameters are automatically set by the given graphs and F-Rank.

Figure 1 indicates that F-Rank is much faster than the original approach. F-Rank cuts the search time from the original approach by up to 40%, 70%, and 90% for P2P, Web, and Wikipedia, respectively; F-Rank can more efficiently find the top-k nodes than the original approach as the graph sizes increase. The original approach iteratively computes PageRank scores until convergence by utilizing the whole graphs which takes $O((N + M)T)$

[1] http://snap.stanford.edu/data/p2p-Gnutella31.html
[2] http://law.di.unimi.it/webdata/cnr-2000/
[3] http://snap.stanford.edu/data/wiki-Talk.html
[4] Even though the original approach does not theoretical guarantee to yield the exact results with this setting (Langville and Meyer 2012), we confirmed that the original approach converges to the exact results with this setting.

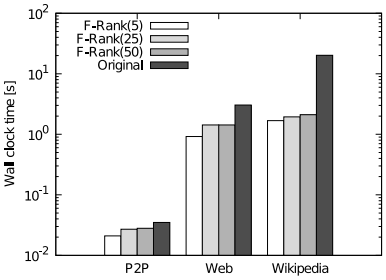


Figure 1: Search time.

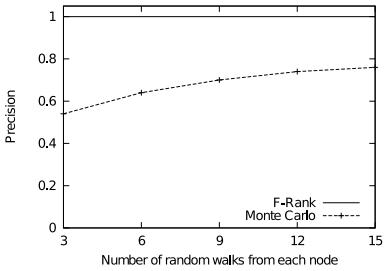


Figure 2: Accuracy versus number of random walks.

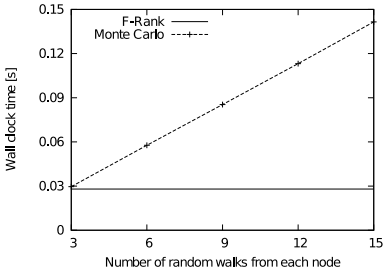


Figure 3: Efficiency versus number of random walks.

time. On the other hand, F-Rank computes the estimations by subgraphs until the number of candidate becomes $k$ which takes $O((n + m + \log c \log k)t)$ time (Theorem 2). As shown in Table 2, in practice, the subgraphs are smaller than the given graphs, and F-Rank needs fewer iterations than the original approach. Additional experiments confirmed that we can reduce the numbers of nodes/edges rapidly in the iterations since the number of candidate nodes monotonically decreases (Lemma 5) and the subgraphs are obtained from the candidate/reachable nodes (Definition 5). As a result, F-Rank is faster than the original approach.

## Exactness

One major advantage of F-Rank is that it outputs the same results as the original approach. To demonstrate this advantage, we compared F-Rank with "MC complete path stopping in dangling nodes" proposed by Avrachenkov et al. (Avrachenkov et al. 2007). Even though they proposed several Monte Carlo approaches, this approach can most effectively approximate PageRank scores among their approaches as reported in their paper. This approach can identify the top-k nodes for a given graph in ad-hoc style by performing random walks several times from each node. Unlike other previous approaches, this approach does not have technical limitations such as unguaranteed score convergence, incremental graph change, or high I/O cost. Since this approach approximates the PageRank score of a node by the total number of visits to the node via the random walks, the number of random walks from each node is expected to impact the search time and approximation accuracy. Therefore, we conducted comparative experiments using various numbers of random walks. Figure 2 and 3 show the accuracy and the search time of each approach for P2P, respectively, where $k = 50$. In Figure 2, we used precision as the metric of accuracy; precision is the fraction of answer nodes that match those of the original approach. We evaluated the efficiency of each approach through wall clock time in Figure 3.

Figure 2 indicates that the precision of F-Rank is 1 since F-Rank guarantees the same results as the original approach (Theorem 1). Figure 2 also shows that precision of the Monte Carlo approach reaches a plateau even though it more accurately finds top-k nodes as the number of random walks increases. The results indicate that the Monte Carlo approach cannot find the top-k nodes exactly while F-Rank outputs only exact answers. Unfortunately, the search time of the Monte Carlo approach is proportional to the number of random walks as shown in Figure 3. These figures show that F-Rank is superior to the Monte Carlo approach in both speed and accuracy.

# Conclusions

This paper proposed an efficient top-k algorithm for PageRank, F-Rank, that guarantees the same results as the original approach. Our algorithm prunes unnecessary nodes and edges by lower/upper estimations and dynamically constructs subgraphs in each iteration to identify the top-k nodes. Experiments showed that our approach has a superiority over the previous approaches. The proposed solution allows many PageRank-based applications to be processed more efficiently, and helps to improve the effectiveness of future applications. There are some avenues for future works. For example, since web-scale graphs can exceed main-memory capacity, disk-aware algorithm is an interesting and challenging problem.

# References


Avrachenkov, K.; Litvak, N.; Nemirovsky, D.; and Osipova, N. 2007. Monte Carlo Methods in PageRank Computation: When One Iteration is Sufficient. *SIAM J. Numerical Analysis*.

Bahmani, B.; Kumar, R.; Mahdian, M.; and Upfal, E. 2012. PageRank on an Evolving Graph. In *KDD*.

Bahmani, B.; Chowdhury, A.; and Goel, A. 2010. Fast Incremental and Personalized PageRank. *PVLDB*.

Bryce, D., and Kambhampati, S. 2007. A Tutorial on Planning Graph Based Reachability Heuristics. *AI Magazine*.

Chien, S.; Dwork, C.; Kumar, R.; Simon, D. R.; and Sivakumar, D. 2003. Link Evolution: Analysis and Algorithms. *Internet Mathematics*.

Cormen, T. H.; Leiserson, C. E.; Rivest, R. L.; and Stein, C. 2009. *Introduction to Algorithms*. The MIT Press.

Dagan, I.; Glickman, O.; Gliozzo, A. M.; Marmorshtein, E.; and Strapparava, C. 2006. Direct Word Sense Matching for Lexical Substitution. In *ACL*.

Erkan, G., and Radev, D. R. 2004. LexRank: Graph-based Lexical Centrality as Salience in Text summarization. *J. Artif. Intell. Res. (JAIR)*.

Fellbaum, C., and Miller, G. 1998. *WordNet: An Electronic Lexical Database*. A Bradford Book.

Fujiwara, Y.; Nakatsuji, M.; Onizuka, M.; and Kitsuregawa, M. 2012a. Fast and Exact Top-k Search for Random Walk with Restart. *PVLDB*.

Fujiwara, Y.; Nakatsuji, M.; Yamamuro, T.; Shiokawa, H.; and Onizuka, M. 2012b. Efficient Personalized PageRank with Accuracy Assurance. In *KDD*.

Fujiwara, Y.; Nakatsuji, M.; Shiokawa, H.; and Onizuka, M. 2013. Efficient Search Algorithm for SimRank. In *ICDE*.

Fujiwara, Y.; Irie, G.; and Kitahara, T. 2011. Fast Algorithm for Affinity Propagation. In *IJCAI*.

Gleich, D.; Zhukov, L.; and Berkhin, P. 2004. Fast Parallel PageRank: A Linear System Approach. Technical report, Yahoo!

Kamvar, S.; Haveliwala, T.; and Golub, G. 2004. Adaptive Methods for the Computation of PageRank. *Linear Algebra and its Applications*. Special Issue on the Conference on the Numerical Solution of Markov Chains.

Khabiri, E.; Caverlee, J.; and Hsu, C.-F. 2011. Summarizing User-contributed Comments. In *International AAAI Conference on Weblogs and Social Media*.

Langville, A. N., and Meyer, C. D. 2006. Updating Markov Chains with an Eye on Google's PageRank. *SIAM J. Matrix Analysis Applications*.

Langville, A. N., and Meyer, C. D. 2012. *Google's PageRank and Beyond: The Science of Search Engine Rankings*. Princeton University Press.

Lee, E.-S.; Yeung, F.; and Yu, T.-Y. 2012. Variable Categorization and Modelling: A Novel Adversarial Approach to mobile location-based advertising. In *AAAI Workshops*.

Manning, C., and Schuetze, H. 1999. *Foundations of Statistical Natural Language Processing*. The MIT Press.

McCarthy, D. 2002. Lexical Substitution as a Task for WSD Evaluation. In *ACL-02 workshop on Word sense disambiguation: recent successes and future directions*.

McSherry, F. 2005. A Uniform Approach to Accelerated PageRank Computation. In *WWW*.

Mehlhorn, K., and Sanders, P. 2010. *Algorithms and Data Structures: The Basic Toolbox*. Springer.

Nakatsuji, M.; Fujiwara, Y.; Uchiyama, T.; and Toda, H. 2012. Collaborative Filtering by Analyzing Dynamic User Interests Modeled by Taxonomy. In *ISWC*.

Page, L.; Brin, S.; Motwani, R.; and Winograd, T. 1999. The PageRank Citation Ranking: Bringing Order to the Web. Technical report, Stanford InfoLab.

Shiokawa, H.; Fujiwara, Y.; and Onizuka, M. 2013. Fast Algorithm for Modularity-based Graph Clustering. In *AAAI*.

Shmueli, E.; Kagian, A.; Koren, Y.; and Lempel, R. 2012. Care to Comment?: Recommendations for Commenting on News Stories. In *WWW*.

Sinha, R., and Mihalcea, R. 2009. Combining Lexical Resources for Contextual Synonym Expansion. In *RANLP*.

Sinha, R. S., and Mihalcea, R. F. 2011. Using Centrality Algorithms on Directed Graphs for Synonym Expansion. In *FLAIRS Conference*.

Watkins, D. S. 2007. *The Matrix Eigenvalue Problem: GR and Krylov Subspace Methods*. Society for Industrial Mathematics.

Yin, X., and Lee, W. S. 2004. Using Link Analysis to Improve Layout on Mobile Devices. In *WWW*.