Quick Detection of Top-k Personalized PageRank Lists

K. Avrachenkov[1], N. Litvak[2], D. Nemirovsky[1], E. Smirnova[1], and M. Sokol[1]

[1] INRIA Sophia Antipolis
{k.avrachenkov,dnemirov,esmirnov,msokol}@sophia.inria.fr
[2] University of Twente
n.litvak@ewi.utwente.nl

Abstract. We study a problem of quick detection of top-k Personalized PageRank (PPR) lists. This problem has a number of important applications such as finding local cuts in large graphs, estimation of similarity distance and person name disambiguation. We argue that two observations are important when finding top-k PPR lists. Firstly, it is crucial that we detect fast the top-k most important neighbors of a node, while the exact order in the top-k list and the exact values of PPR are by far not so crucial. Secondly, by allowing a small number of "wrong" elements in top-k lists, we achieve great computational savings, in fact, without degrading the quality of the results. Based on these ideas, we propose Monte Carlo methods for quick detection of top-k PPR lists. We demonstrate the effectiveness of these methods on the Web and Wikipedia graphs, provide performance evaluation and supply stopping criteria.

1 Introduction

Personalized PageRank (PPR) or Topic-Sensitive PageRank [15] is a generalization of PageRank [10], and is a stationary distribution of a random walk on an entity graph, with random restart from a given personalization distribution. Originally designed for personalization of the Web search results [15], PPR found a large number of network applications, e.g., in finding related entities [11], graph clustering and finding local cuts [1, 4], link predictions in social networks [20] and protein-protein interaction networks [23]. The recent application of PPR to the person name disambiguation problem lead to the first official place in the WePS 2010 challenge [21]. In most of applications, e.g., in name disambiguation, one is mainly interested in detecting top-k elements with the largest PPR. This work on detecting top-k elements is driven by the following two key observations:

Observation 1: Often it is extremely important to detect fast the top-k elements with the largest PPR, while the exact order in the top-k list as well as the exact values of the PPR are by far not so important. Application examples are given in the above mentioned references.

Observation 2: We may apply a relaxation that allows a small number of elements to be placed erroneously in the top-k list. If the PPR values of these elements are of a similar order of magnitude as in the top-k list, then such relaxation does not affect applications, but it enables us to take advantage of the generic "80/20 rule": 80% of the result is achieved with 20% of efforts.

We argue that the Monte Carlo approach naturally takes into account the two key observations. In [9] this approach was proposed for the computation of the standard PageRank. The estimation of the convergence rate in [9] was very pessimistic. The implementation of the Monte Carlo approach was improved in [13] and also applied there to PPR. Both [9] and [13] only use end points of the random walks to compute the PageRank values. Moreover, [13] requires extensive precomputation efforts and is very demanding in storage resource. In [5] the authors have further improved the realization of the Monte Carlo method [13]. In [2] it is shown that Monte Carlo estimation for large PageRank values requires about the same number of operations as one iteration of the power iteration method. In this paper we show that the Monte Carlo algorithms require an incomparably smaller number of operations when our goal is to detect a top-k list with k not large. In our test on the Wikipedia entity graph with about 2 million nodes typically few thousands of operations are enough to detect the top-10 list with just two or three erroneous elements. Hence, we obtain a relaxation of the top-10 list with just about 1-5% of operations required by one power iteration. Experimental results on the Web graph appear to be even more striking. In the present work we provide theoretical justifications for such remarkable efficiency. We would like to emphasize that the Monte Carlo approach allows easy online and parallel implementation and does not require the knowledge of the complete graph.

We consider the present work as an initiation to a new line of research on quick detection of top-k ranked network central elements. A number of interesting questions will be addressed in the future research: What is the difference in performance between the randomized algorithms, like the presented Monte Carlo algorithms, and the non-randomized algorithms, like algorithms in [1] and [7]? What are efficient practical stopping criteria for the randomized algorithms? What is the effect of the graph structure on the performance of the randomized algorithms?

2 Monte Carlo methods

Given a directed or undirected graph connecting some entities, the PPR $\pi(s,c)$ with a seed node s and a damping parameter c is defined as a solution of the following equations

$$\pi(s,c) = c\pi(s,c)P + (1-c)\mathbf{1}_s^T, \qquad \sum_{j=1}^{n} \pi_j(s,c) = 1,$$

where $\mathbf{1}_s^T$ is a row unit vector with one in the s^{th} entry and all the other elements equal to zero, P is the transition matrix associated with the entity graph and n

is the number of entities. Equivalently, PPR can be given by [19]

$$\pi(s,c) = (1-c)\mathbf{1}_s^T[I-cP]^{-1}. \tag{1}$$

When the values of s and c are clear from the context we shall simply write π.

We note that PPR is often defined with a general distribution v in place of $\mathbf{1}_s^T$. However, typically v has a small support. Then, due to linearity, the problem of PPR with distribution v reduces to computing PPR with distribution $\mathbf{1}_s^T$ [16].

In this work we consider two Monte Carlo algorithms. The first algorithm is inspired by the following observation. Consider a random walk $\{X_t\}_{t\geq 0}$ that starts from node s, i.e, $X_0 = s$. Let at each step the random walk terminate with probability $1-c$ and make a transition according to the matrix P with probability c. Then, the end-points of such a random walk has the distribution $\pi(s,c)$.

Algorithm 1 (MC End Point) *Simulate m runs of the random walk $\{X_t\}_{t\geq 0}$ initiated at node s. Evaluate π_j as a fraction of m random walks which end at node $j \in 1,\ldots,n$.*

Next, we exploit the fact that the element (s,j) of the matrix $[I-cP]^{-1}$ equals to the expected number of visits to node j by the random walk initiated at state s with the run time geometrically distributed with parameter c [2]. Thus, the formula (1) suggests the following estimator for the PPR

$$\hat{\pi}_j(s,c) = (1-c)\frac{1}{m}\sum_{r=1}^{m} N_j(s,r), \tag{2}$$

where $N_j(s,r)$ is the number of visits to state j during the run r of the random walk initiated at node s. This leads to our second Monte Carlo algorithm.

Algorithm 2 (MC Complete Path) *Simulate m runs of the random walk $\{X_t\}_{t\geq 0}$ initiated at node s. Evaluate π_j as the total number of visits to node j multiplied by $(1-c)/m$.*

As outputs of the proposed algorithms we would like to obtain with high probability either a *top-k list* of nodes or a *top-k basket* of nodes.

Definition 1. *The* top-k list *of nodes is a list of k nodes with largest PPR values arranged in a descending order of their PPR values.*

Definition 2. *The* top-k basket *of nodes is a set of k nodes with largest PPR values with no ordering required.*

It turns out that it is beneficial to relax our goal and to obtain a top-k basket with a small number of erroneous elements.

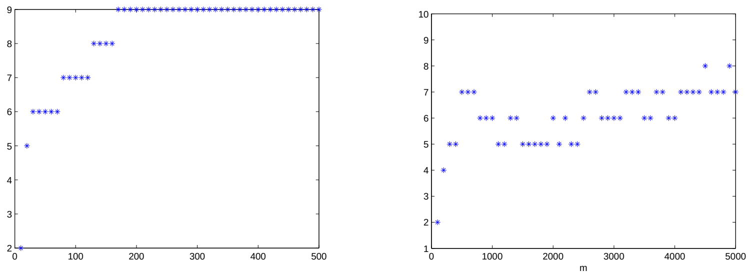
Definition 3. *We call* relaxation-l *top-k basket a realization when we allow at most l erroneous elements from top-k basket.*

In the present work we aim to estimate the numbers of random walk runs m sufficient for obtaining top-k list or top-k basket or relaxation-l top-k basket with high probability. In particular, we demonstrate that ranking converges considerably faster than the values of PPR and that a relaxation-l with quite small l helps significantly.

Throughout the paper we illustrate the theoretical analysis with the help of experiments on two large graphs: the Wikipedia entity graph and the Web graph. There is a number of reasons why we have chosen the Wikipedia entity graph. Firstly, all elements of PPR can be computed with high precision for the Wikipedia entity graph with the help of BVGraph/WebGraph framework [8]. Secondly, the Wikipedia graph has already been used in several applications related to finding top-k semantically related entities. Thirdly, since the Wikipedia entity graph has a very small average distance [24], it represents a very challenging test for the Monte Carlo methods. In just 3-4 steps the random walk can be very far from the starting node. Since the Monte Carlo approach does not require the knowledge of a complete graph, we can apply our algorithms to the actual Web graph. However, computing the exact values for the Pesonalized PageRank of web pages is infeasible in our experiments. We can only obtain correct top-k lists by Monte Carlo methods with very high probability as in [2, 13] using an ample number of crawls.

Illustrating example with Wikipedia: Following our recent work [21] we illustrate PPR by application to the person name disambiguation problem. One of the most common English names is Jackson. We have selected three Jacksons who have entries in Wikipedia: Jim Jackson (ice hockey), Jim Jackson (sportscaster) and Michael Jackson. Two Jacksons have even a common given name and both worked in ice hockey, one as an ice hockey player and another as an ice hockey sportscaster. In [3] we provide the exact lists of top-10 Wikipedia articles arranged according to PPR vectors. We observe that an exact top-10 list identifies quite well its seed node. Next, we run the Monte Carlo End Point method starting from each seed node. Notice that to obtain a relaxed top-10 list with two or three erroneous elements we need different number of runs for different seed nodes (50000 runs for `Michael Jackson` vs. 500 runs for `Jim Jackson (ice hockey)`). Intuitively, the more immediate neighbours a node has, the larger number of Monte Carlo steps is required. Indeed, if a seed node has many immediate neighbours then the Monte Carlo method easily drifts away. In Figures 1-2.(a) we present examples of typical runs of the Monte Carlo End Point method for the three different seed nodes. An example of the Monte Carlo Complete Path method for the seed node `Michael Jackson` is given in Figure 2.(b). As expected, it outperforms the Monte Carlo End Point method. In the following sections we shall quantify all the above qualitative observations.

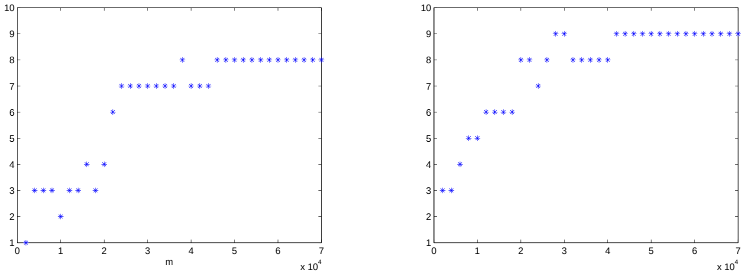
Illustrating example with the Web: We have also tested our two Monte Carlo methods on the Web. To see the difference in comparison with a "smaller" Wikipedia graph we have chosen the official Web page of Michael Jackson `http://www.michaeljackson.com` and the Web page of the hockey player statistics Jim Jackson hosted at `http://www.hockeydb.com`. In Figures 3.(a) and 3.(b) we present examples of typical runs of the Monte Carlo Complete Path and End Point methods for, respectively, the Michael Jackson Web page and the Jim Jackson Web page as a seed node. We have performed enough steps (6×10^5) to make sure that the top-k lists of nodes are stabilized so that we could say with very high certainty that we know the correct top-k lists. We observe that in comparison to the Wikipedia graph we need longer runs. However, the amount of computational saving is still very impressive. Indeed, according to even modest estimates, the size of the Web is more than 10^{10} pages. However, to get a good top-k list for the Michael Jackson page we need about 10^5 steps with MC Complete Path. Thus, we are using only 10^{-5} fraction of computational resources which are needed for just one power iteration!



(a) Seed node `Jim Jackson (ice hockey)`.

(b) Seed node `Jim Jackson (sportscaster)`.

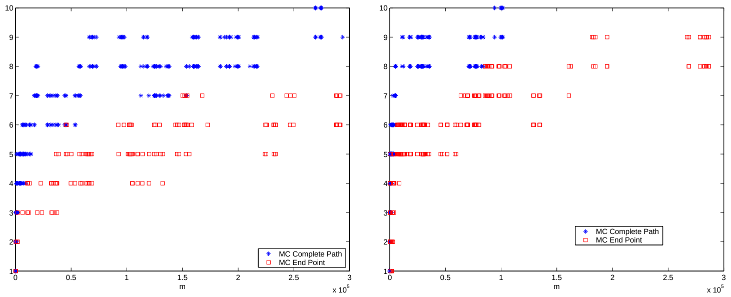
Fig. 1. The number of correctly detected elements by MC End Point.



(a) Seed node `Michael Jackson`.

(b) Seed node `Michael Jackson`.

Fig. 2. The number of correctly detected elements by MC End Point (a) and MC Complete Path (b).



(a) Seed node Michael Jackson Web page.

(b) Seed node Jim Jackson Web page.

Fig. 3. The number of correctly detected elements by MC End Point.

3 Variance based performance comparison and CLT approximations

In the MC End Point algorithm the distribution of end points is multinomial [17]. Namely, if we denote by L_j the number of paths that end at node j after m runs, then we have

$$P\{L_1 = l_1, L_2 = l_2, \ldots, L_n = l_n\} = \frac{m!}{l_1! l_2! \cdots l_n!} \pi_1^{l_1} \pi_2^{l_2} \cdots \pi_n^{l_n}. \qquad (3)$$

Thus, the standard deviation of the MC End Point estimator for the k^{th} element is given by

$$\sigma(\hat{\pi}_k) = \sigma(L_k/m) = \frac{1}{\sqrt{m}} \sqrt{\pi_k(1-\pi_k)}. \qquad (4)$$

An expression for the standard deviation of the MC Complete Path is more complicated. Define the matrix $Z = (z_{ij}) = [I - cP]^{-1}$ and let N_j be the number of visits to node j by the random walk with the run time geometrically distributed with parameter c. Further, denote by $E_i(\cdot)$ a conditional expectation provided that the random walk starts at $i = 1, \ldots, n$. From (2), it follows that

$$\sigma(\hat{\pi}_k) = \frac{(1-c)}{\sqrt{m}} \sigma(N_k) = \frac{(1-c)}{\sqrt{m}} \sqrt{E_s\{N_k^2\} - E_i\{N_k\}^2}. \qquad (5)$$

First, we recall that

$$E_s\{N_k\} = z_{sk} = \pi_k(s)/(1-c). \qquad (6)$$

Then, from [18], it is known that $E_s\{N_k^2\} = [Z(2Z_{dg} - I)]_{sk}$, where Z_{dg} is a diagonal matrix having as its diagonal the diagonal of matrix Z and $[A]_{ik}$ is the $(i,k)^{\text{th}}$ element of matrix A. Thus, we write

$$
\begin{aligned}
E_s\{N_k^2\} &= \mathbf{1}_s^T Z(2Z_{dg} - I)\mathbf{1}_k = \frac{1}{1-c}\pi(s)(2Z_{dg} - I)\mathbf{1}_k \\
&= \frac{1}{1-c}\left(\frac{1}{1-c}\pi_k(s)\pi_k(k) - \pi_k(s)\right). \qquad (7)
\end{aligned}
$$

Substituting (6) and (7) into (5), we obtain

$$
\sigma(\hat{\pi}_k) = \frac{1}{\sqrt{m}}\sqrt{\pi_k(s)(2\pi_k(k) - (1-c) - \pi_k(s))}. \qquad (8)
$$

Since $\pi_k(k) \approx 1 - c$, we can approximate $\sigma(\hat{\pi}_k)$ with

$$
\sigma(\hat{\pi}_k) \approx \frac{1}{\sqrt{m}}\sqrt{\pi_k(s)((1-c) - \pi_k(s))}.
$$

Comparing the latter expression with (4), we see that MC End Point requires approximately $1/(1-c)$ walks more than MC Complete Path. This was expected as MC End Point uses only information from end points of the random walks. We would like to emphasize that $1/(1-c)$ can be a significant coefficient. For instance, if $c = 0.85$, then $1/(1-c) \approx 6.7$.

Now, for the MC End Point we can use CLT-type result given e.g. in [22]:

Theorem 1. *[22] For large m and $\sum_{i=1}^n l_i = m$, a multivariate normal density approximation to the multinomial distribution* (3) *is given by*

$$
f(l_1, l_2, \ldots, l_n) = \left(\frac{1}{2\pi m}\right)^{(n-1)/2} \times \left(\frac{1}{n\pi_1\pi_2\cdots\pi_n}\right)^{1/2} \exp\left\{-\frac{1}{2}\sum_{i=1}^n \frac{(l_i - m\pi_i)^2}{m\pi_i}\right\}. \qquad (9)
$$

For the MC Complete Path, we note that $N(s,r) = (N_1(s,r), \ldots, N_n(s,r))$, $r = 1, 2, \ldots$, form a sequence of i.i.d. random vectors. Hence, we can apply the multivariate central limit theorem. Denote

$$
\hat{N}(s,m) = \frac{1}{m}\sum_{r=1}^m N(s,r). \qquad (10)
$$

Theorem 2. *Let m go to infinity. Then, we have the following convergence in distribution to a multivariate normal distribution*

$$
\sqrt{m}\left(\hat{N}(s,m) - \bar{N}\right) \xrightarrow{D} \mathcal{N}(0, \Sigma(s)),
$$

where $\bar{N}(s) = \mathbf{1}_s^T Z$ and $\Sigma(s) = E\{N^T(s,r)N(s,r)\} - \bar{N}^T(s)\bar{N}(s)$ is a covariance matrix, which can be expressed as

$$
\Sigma(s) = \Omega(s)Z + Z^T\Omega(s) - \Omega(s) - Z^T\mathbf{1}_s\mathbf{1}_s^T Z. \qquad (11)
$$

where the matrix $\Omega(s) = \{\omega_{jk}(s)\}$ is defined by

$$
\omega_{jk}(s) = \begin{cases} z_{sj}, \text{ if } j = k, \\ 0, \quad \text{otherwise.} \end{cases}
$$

Proof. See [3].

We would like to note that in both cases we obtain the convergence to rank deficient (singular) multivariate normal distributions.
Illustrating example with the Web (cont.): In Table 1 we provide means and standard deviations for the number of hits of MC End Point for top-10 nodes with the Jim Jackson Web page as the seed node for the number of runs $m = 10^4$ and $m = 10^5$. We observe that the means are very close to each other and the standard deviations are significant with respect to the values of the means. This shows that a direct application of the central limit theorem and the confidence intervals technique will lead to inadequate stopping criteria. In the ensuing sections we discuss metrics and stopping criteria which are much more efficient for the present problem.

Table 1. MC End Point for the Jim Jackson Web page: means and Standard Deviations

nr. runs	10000		100000	
rank	mean	std	mean	std
1	0.79104	0.012531	0.79748	0.004834
2	0.006329	0.001622	0.006202	0.000569
3	0.005766	0.001075	0.005885	0.000354
4	0.006365	0.002103	0.006561	0.000704
5	0.005183	0.001664	0.005518	0.00055
6	0.005541	0.003766	0.005801	0.001257
7	0.007617	0.003633	0.006243	0.001266
8	0.007566	0.012384	0.005854	0.003969
9	0.006186	0.001468	0.006182	0.000547
10	0.00672	0.003492	0.006223	0.001132

4 Convergence based on order

For the two introduced Monte Carlo methods we aim to calculate or estimate a probability that after a given number of steps we correctly obtain top-k list or top-k basket. These are the probabilities $P\{L_1 > \cdots > L_k > L_j, \forall j > k\}$ and $P\{L_i > L_j, \forall i,j : i \le k < j\}$ respectively, where L_k, $k \in 1,\dots,n$, can be either the Monte Carlo estimates or the ranked elements or their CLT approximations. We refer to these probabilities as the ranking probabilities and we refer to complementary probabilities as misranking probabilities [6]. Because of combinatorial explosion, exact calculation of these probabilities is infeasible in non-trivial cases. Thus, we propose estimation methods based on Bonferroni inequality. This approach works for reasonably large values of m.

Drawing correctly the top-k basket is defined by the event $\bigcap_{i \le k < j}\{L_i > L_j\}$. Applying the Bonferroni inequality $P\left\{\bigcap_s A_s\right\} \ge 1 - \sum_s P\left\{\bar{A}_s\right\}$ to this event, we obtain $P\left\{\bigcap_{i \le k < j}\{L_i > L_j\}\right\} \ge 1 - \sum_{i \le k < j} P\left\{\overline{\{L_i > L_j\}}\right\}$. Equivalently, we can write the following upper bound for the misranking probability

$$1 - P\left\{\bigcap_{i \le k < j}\{L_i > L_j\}\right\} \le \sum_{i \le k < j} P\left\{L_i \le L_j\right\}. \qquad (12)$$

We note that the upper bound for the misranking probability is very useful, because it will provide a guarantee on the performance of our algorithms. Since in the MC End Point method the distribution of end points is multinomial (see (3)), for small m we can directly use the formula

$$P\{L_i \leq L_j\} = \sum_{l_i+l_j\leq m,\ l_i\leq l_j} \frac{m!}{l_i!l_j!(m-l_i-l_j)!}\pi_i^{l_i}\pi_j^{l_j}(1-\pi_i-\pi_j)^{m-l_i-l_j}. \quad (13)$$

For large m it is computationally intractable. Hence, we now turn to the CLT approximations for the both MC methods. Denote by L_j the original number of hits at node j and by Y_j its CLT approximation. First, we obtain a CLT based expression for the misranking probability for two nodes $P\left\{Y_i \leq Y_j\right\}$. Since the event $\{Y_i \leq Y_j\}$ coincides with the event $\{Y_i - Y_j \leq 0\}$ and a difference of two normal random variables is again a normal random variable, we obtain $P\left\{Y_i \leq Y_j\right\} = P\left\{Y_i - Y_j \leq 0\right\} = 1 - \Phi(\sqrt{m}\rho_{ij})$, where $\Phi(\cdot)$ is the cumulative distribution function for the standard normal random variable and

$$\rho_{ij} = \frac{E[Y_i] - E[Y_j]}{\sqrt{\sigma^2(Y_i) - 2\mathrm{cov}(Y_i, Y_j) + \sigma^2(Y_j)}}.$$

For large m, the above expression can be bounded by $P\left\{Y_i \leq Y_j\right\} \leq \frac{1}{\sqrt{2\pi}}e^{-\frac{\rho_{ij}^2}{2}m}$. Since the misranking probability for two nodes $P\left\{Y_i \leq Y_j\right\}$ decreases when j increases, we can write

$$1 - P\left\{\bigcap_{i\leq k<j}\{Y_i > Y_j\}\right\} \leq \sum_{i=1}^{k}\left(\sum_{j=k+1}^{j^*} P\left\{Y_i \leq Y_j\right\} + \sum_{j=j^*+1}^{n} P\left\{Y_i \leq Y_{j^*}\right\}\right),$$

for some j^*. This gives the following upper bound

$$1-P\left\{\bigcap_{i\leq k<j}\{Y_i > Y_j\}\right\} \leq \sum_{i=1}^{k}\sum_{j=k+1}^{j^*}(1-\Phi(\sqrt{m}\rho_{ij})) + \frac{n-j^*}{\sqrt{2\pi}}\sum_{i=1}^{k} e^{-\frac{\rho_{ij^*}^2}{2}m}. \quad (14)$$

Since we have a finite number of terms in the right hand side of expression (14), we conclude that

Theorem 3. *The misranking probability of the top-k basket goes to zero with geometric rate,* $1 - P\left\{\bigcap_{i\leq k<j}\{Y_i > Y_j\}\right\} \leq Ca^m$, *for some* $C > 0$, $a \in (0, 1)$.

We note that the multinomial distribution, ρ_{ij} has a simple expression

$$\rho_{ij} = \frac{\pi_i - \pi_j}{\sqrt{\pi_i(1-\pi_i) + 2\pi_i\pi_j + \pi_j(1-\pi_j)}}.$$

For MC Complete Path $\sigma^2(Y_i) = \Sigma_{ii}(s)$ and $\mathrm{cov}(Y_i, Y_j) = \Sigma_{ij}(s)$ where $\Sigma_{ii}(s)$ and $\Sigma_{ij}(s)$ can be calculated by (11). Similarly the Bonferroni inequality can be applied to the top-k list (see [3]).

5 Solution relaxation

In this section we analytically evaluate the relation between the number of experiments m and the average number of correctly identified top-k nodes. We use the relaxation by allowing the latter number to be smaller than k. We aim to mathematically justify the observed "80/20 behavior" of the algorithm: 80 percent of the top-k nodes are identified correctly in a very short time.

Let M_0 be a number of correctly identified elements in the top-k basket. In addition, denote by K_i the number of nodes ranked not lower than i. Formally, $K_i = \sum_{j \neq i} 1\{L_j \geq L_i\}, \quad i = 1, \ldots, k$, where $1\{\cdot\}$ is an indicator function. Placing node i in the top-k basket is equivalent to the event $\{K_i < k\}$, and thus $E(M_0) = E\left(\sum_{i=1}^{k} 1\{K_i < k\}\right) = \sum_{i=1}^{k} P(K_i < k)$. Direct evaluation of $P(K_i < k)$ is computationally intractable in realistic scenarios, even with Markov chain representation techniques [12]. Thus, we use *approximation* and *Poissonisation*.

The End Point algorithm is merely an occupancy scheme where each independent experiment (random walk) results in placing one ball (visit) to an urn (node of the graph). Under Poissonisation [14], we assume that the number of random walks is a Poisson random variable M with given mean m. Because the number of hits in the Poissonised model is different from the number of original hits, we use the notation Y_i instead of L_j for the number of visits to page j. Note that Y_j is a Poisson random variable with parameter $m\pi_j$ and is independent of Y_i for $i \neq j$. The imposed independence of Y_j's greatly simplifies the analysis.

Next to Poissonisation, we also apply *approximation* of M_0 by a closely related measure M_1: $M_1 = k - \sum_{i=1}^{k}(K'_i/k)$, where K'_i denotes the number of pages outside the top-k list that are ranked higher than node $i = 1, \ldots, k$. Note that K'_i is the number of mistakes with respect to node i that lead to errors in the identified top-k list. Then the sum in the definition of M_1 is simply the average number of such mistakes with respect to each of the top-k nodes.

The measure M_1 is more tractable than M_0 because its average value $E(M_1) = k - \frac{1}{k}\sum_{i=1}^{k} E(K'_i)$ involves only the average values of K'_i and not their distributions, and because K'_i depends only on the nodes outside the top-k list. Then, we can make use of the following convenient measure $\mu(y)$:

$$\mu(y) := E(K'_i | Y_i = y) = \sum_{j=k+1}^{n} P(Y_j \geq y), \; i = 1, \ldots, k,$$

which implies $E(K'_i) = \sum_{y=0}^{\infty} P(Y_i = y)\mu(y), \; i = 1, \ldots, k$. Therefore, we obtain the following expression for $E(M_1)$:

$$E(M_1) = k - \frac{1}{k}\sum_{y=0}^{\infty} \mu(y) \sum_{i=1}^{k} P(Y_i = y). \tag{15}$$

Illustrating example with Wikipedia (cont.): Let us calculate $E(M_1)$ for the top-10 basket corresponding to the seed node `Jim Jackson (ice hockey)`. Using formula (15), for $m = 8 \times 10^3; 10 \times 10^3; 15 \times 10^3$ we obtain $E(M_1) =$

$7.75; 9.36; 9.53$. It took 2000 runs to move from $E(M_1) = 7.75$ to $E(M_1) = 9.36$, but then 5000 runs is needed to advance from $E(M_1) = 9.36$ to $E(M_1) = 9.53$. We see that we obtain quickly 2-relaxation or 1-relaxation of the top-10 basket but then we need to spend a significant amount of effort to get the complete basket. This is indeed in agreement with the Monte Carlo runs (see e.g., Figure 1). In the next theorem we explain this "80/20 behavior" and provide indication for the choice of m.

Theorem 4. *In the Poisonized End Point Monte Carlo algorithm, if all top-k nodes receive at least $y = ma > 1$ visits and $\pi_{k+1} = (1-\varepsilon)a$, $\varepsilon > 1/y$, then*

(i) to satisfy $E(M_1) > (1-\alpha)k$ it is sufficient to have

$$\sum_{j=k+1}^{n} \frac{(m\pi_j)^y}{y!} e^{-m\pi_j} \left[1 + \sum_{l=1}^{\infty} \frac{(m\pi_j)^l}{(y+1)\cdots(y+l)} \right] < \alpha k.$$

(ii) Statement (i) is always satisfied if $m > 2a^{-1}\varepsilon^{-2}[-\log(\varepsilon\pi_{k+1}\alpha k)]$.

Proof. See [3].

From (i) we can already see that the 80/20 behavior of $E(M_1)$ (and, respectively, $E(M_0)$) can be explained mainly by the fact that $\mu(y)$ drops drastically with y because the Poisson probabilities decrease faster than exponentially.

The bound in (ii) shows that m should be roughly of the order $1/\pi_k$. The term ε^{-2} is not defining since ε does not need to be small. For instance, by choosing $\varepsilon = 1/2$ we can filter out the nodes with PPR not higher than $\pi_k/2$. This often may be sufficient in applications. Obviously, the logarithmic term is of a smaller order of magnitude.

We note that the bound in (ii) is quite rough because in its derivation (see [3]) we replaced π_j, $j > k$, by their maximum value π_{k+1}. In realistic examples, m can be chosen much smaller than in (ii) of Theorem 4. In fact, in our examples good top-k baskets are obtained if the algorithm is terminated at the point when for some y, each node in the current top-k basket has received at least y visits while the rest of the nodes have received at most $y - d$ visits, where d is a small number, say $d = 2$. Such choice of m satisfies (i) with reasonably small α. Without a formal justification, this stopping rule can be understood since we have $m\pi_{k+1} = ma(1-\varepsilon) \approx ma - d$, which results in a small value of $\mu(y)$.

Acknowledgments: We would like to thank Brigitte Trousse for her very helpful remarks and suggestions.

References

1. R. Andersen, F. Chung and K. Lang, "Local graph partitioning using pagerank vectors", in Proceedings of FOCS 2006, pp.475-486.
2. 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*, v.45, no.2, pp.890-904, 2007.

3. K. Avrachenkov, N. Litvak, D. Nemirovsky, E. Smirnova and M. Sokol, "Monte Carlo Methods for Top-k Personalized PageRank Lists and Name Disambiguation", INRIA Research Report no.7367, 2010.
4. K. Avrachenkov, V. Dobrynin, D. Nemirovsky, S.K. Pham, and E. Smirnova, "PageRank Based Clustering of Hypertext Document Collections", in Proceedings of ACM SIGIR 2008, pp.873-874.
5. B. Bahmani, A. Chowdhury and A. Goel, "Fast Incremental and Personalized PageRank", in Proceedings of VLDB Endow., 2010.
6. C. Barakat, G. Iannaccone, and C. Diot, "Ranking flows from sampled traffic", in Proceedings of CoNEXT 2005.
7. P. Berkhin, "Bookmark-Coloring Algorithm for Personalized PageRank Computing", *Internet Mathematics*, v.3, pp.41-62, 2006.
8. P. Boldi and S. Vigna, "The WebGraph framework I: Compression techniques", in Proceedings of the 13th International World Wide Web Conference (WWW 2004)", pp.595-601, 2004.
9. L.A. Breyer, "Markovian Page Ranking distributions: Some theory and simulations", Technical Report 2002, available at http://www.lbreyer.com/preprints.html.
10. S. Brin, L. Page, R. Motwami, and T. Winograd, "The PageRank citation ranking: bringing order to the Web", Stanford University Technical Report, 1998.
11. S. Chakrabarti, "Dynamic Personalized PageRank in entity-relation graphs", in Proceedings of WWW2007.
12. C.J. Corrado, "The exact joint distribution for the multinomial maximum and minimum and the exact distribution for the multinomial range", SSRN Research Report, 2007.
13. D. Fogaras, B. Rácz, K. Csalogány and T. Sarlós, "Towards scaling fully personalized Pagerank: Algorithms, lower bounds, and experiments", *Internet Mathematics*, v.2(3), pp.333-358, 2005.
14. A. Gnedin, B. Hansen and J. Pitman, "Notes on the occupancy problem with infinitely many boxes: general asymptotics and power laws", *Probability Survyes*, v.4, pp.146-171, 2007.
15. T. Haveliwala, "Topic-Sensitive PageRank", in Proceedings of WWW2002.
16. G. Jeh and J. Widom,"Scaling personalized web search", in Proceedings of WWW 2003.
17. K.L. Johnson, S. Kotz and N. Balakrishnan, *Discrete Multivariate Distributions*, Wiley, New York, 1997.
18. J. Kemeny and J. Snell, *Finite Markov Chains*, Springer, 1976.
19. A.N. Langville and C.D. Meyer, *Google's PageRank and Beyond: The Science of Search Engine Rankings*, Princeton University Press, 2006.
20. D. Liben-Nowell and J. Kleinberg, "The link prediction problem for social networks", in Proceedings of CIKM 2003.
21. E. Smirnova, K. Avrachenkov and B. Trousse, "Using Web Graph Structure for Person Name Disambiguation", in Proceedings of CLEF/WEPS 2010.
22. K. Tanabe and M. Sagae, "An exact Cholesky Decomposition and the generalized inverse of the variance-covariance matrix of the multinomial distribution, with applications", *Journal of the Royal Statistical Society (Series B)*, v.54, no.1, pp.211-219, 1992.
23. K. Voevodski, S.H. Teng and Y. Xia, "Spectral affinity in protein networks", *BMC Systems Biology* 3:112, 2009.
24. V. Zlatic, M. Bozicevic, H. Stefancic and M. Domazet, "Wikipedias: Collaborative web-based encyclopedias as complex networks", *Phys. Rev. E*, v.74, 2006.