# Efficiently Handling Dynamics in Distributed Link Based Authority Analysis

Josiane Xavier Parreira[1], Sebastian Michel[2], and Gerhard Weikum[1]

[1] Max-Planck Institute for Informatics
Saarbrücken, Germany
{jparreir,weikum}@mpi-inf.mpg.de
[2] Ecole Polytechnique Fédérale de Lausanne
Lausanne, Switzerland
sebastian.michel@epfl.ch

**Abstract.** Link based authority analysis is an important tool for ranking resources in social networks and other graphs. Previous work have presented $\mathrm{J}_\mathrm{P}^\mathrm{X}$, a decentralized algorithm for computing PageRank scores. The algorithm is designed to work in distributed systems, such as peer-to-peer (P2P) networks. However, the dynamics of the P2P networks, one if its main characteristics, is currently not handled by the algorithm. This paper shows how to adapt $\mathrm{J}_\mathrm{P}^\mathrm{X}$ to work under network churn. First, we present a distributed algorithm that estimates the number of distinct documents in the network, which is needed in the local computation of the PageRank scores. We then present a method that enables each peer to detect other peers leave and to update its view of the network. We show that the number of stored items in the network can be efficiently estimated, with little overhead on the network traffic. Second, we present an extension of the original $\mathrm{J}_\mathrm{P}^\mathrm{X}$ algorithms that can cope with network and content dynamics. We show by a comprehensive performance analysis the practical usability of our approach. The proposed estimators together with the changes in the core $\mathrm{J}_\mathrm{P}^\mathrm{X}$ components allow for a fast and authority score computation even under heavy churn. We believe that this is the last missing step toward the application of distributed PageRank measures in real-life large-scale applications.

## 1 Introduction

The peer-to-peer (P2P) approach facilitates the sharing of huge amounts of data in a distributed and self-organizing way. These characteristics offer enormous potential benefit for search capabilities powerful in terms of scalability, efficiency, and resilience to failures and dynamics. Additionally, a P2P search engine can potentially benefit from the intellectual input (e.g., bookmarks, query logs, click streams, etc.) of a large user community participating in the data sharing network. Finally, but perhaps even more importantly, a P2P search engine can also facilitate pluralism in informing users about Internet content, which is crucial in order to preclude the formation of information-resource monopolies and the biased visibility of content from economically powerful sources.

A conceivable, very intriguing application of P2P computing is Web search. The functionality would include search for names and simple attributes of files, but also

Google-style keyword or even richer XML-oriented search capabilities. It is important to point out that Web search is not simply keyword filtering, but involves relevance assessment and ranking search results. We envision an architecture where each peer can compile its data at its discretion, according to the user's personal interests and data production activities (e.g., publications, blogs, news gathered from different feeds, Web pages collected by a thematically focused crawl). Queries can be executed locally on the small-to-medium personalized corpus, but they can also be forwarded to other, appropriately selected, peers for additional or better search results.

In previous works [25,26], we have presented the $\mathrm{J}_{\mathrm{P}}^{\mathrm{X}}$ algorithm for decentralized computation of global PageRank scores in a P2P network. It works by combining local PageRank computation at peers and exchange of messages in the network. The authority scores obtained with $\mathrm{J}_{\mathrm{P}}^{\mathrm{X}}$ are proved to converge to the global PageRank scores that one would obtain by running the PageRank algorithm in the union of all contents in the network.

However the algorithm currently does not handle one of the main characteristics of P2P networks, namely their dynamic nature. Peers are constantly joining and leaving the network, meaning that the fully content is not always available. Moreover, peers might change what they store, for instance a user can become interested in a different topic and start to store information about this new topic instead. This has a big impact on the computation of authority scores, since the endorsement links might as well change.

In this work we propose methods to adapt the $\mathrm{J}_{\mathrm{P}}^{\mathrm{X}}$ algorithm to work under dynamics. The dynamics considered here can be of one of the two types: network dynamics and content dynamics. Network dynamics refers to changes on the peer population since nodes are continuously joining and leaving the system. Content dynamics refers to changes on the what is stored by the peers.

This paper is organized as follows. Section 3 briefly reviews the basic principles and properties of $\mathrm{J}_{\mathrm{P}}^{\mathrm{X}}$. Section 4 shows how global statistics can be gathered using small statistical sketches that are piggy-backed onto the existing communication. These statistics are of fundamental importance to adjust the algorithm to work under dynamics. Section 5 addresses exactly these countermeasures to avoid inaccurate PageRank estimations when the underlying data changes due to node failures/departures or changing data. Section 6 presents the experimental results. Section 7 concludes the paper and gives an overview on ongoing and future work.

## 2 Related Work

Link-based authority ranking has received great attention in the literature. Good surveys of the many improvements and variations are given in [9,20,7,5].

The basic idea of PageRank [8] is that if page $p$ has a link to page $q$ then the author of $p$ is implicitly endorsing $q$, i.e., giving some importance to page $q$. How much $p$ contributes to the importance of $q$ is proportional to the importance of $p$ itself.

This recursive definition of importance is captured by the stationary distribution of a Markov chain that describes a random walk over the graph, where we start at an arbitrary page and in each step choose a random outgoing edge from the current page. To ensure the ergodicity of the chain, random jumps among pages are allowed, with smaller probability.

With the advent of P2P networks [1,31,27,28] a lot of research has been dedicated to distributed link analysis techniques has been growing.

In [32] Wang and DeWitt presented a distributed search engine framework, in which the authority score of each page is computed by performing the PageRank algorithm at the Web server that is the responsible host for the page, based only on the intra-server links. They also assign authority scores to each server in the network, based on the inter-server links, and then approximate global PageRank values by combining local page authority scores and server authority values. Wu and Aberer [33] pursue a similar approach based on a layered Markov model. Both of these approaches are in turn closely related to the work by Haveliwala et al. [17] that postulates a block structure of the link matrix and exploits this structure for faster convergence of the global PageRank computation.

Other techniques [19,11] for approximating PageRank-style authority scores with partial knowledge of the global graph use state-aggregation technique from the stationary analysis of large Markov chains. A storage-efficient approach to computing authority scores is the OPIC algorithm developed by Abiteboul et al. [2]. This method avoids having the entire link graph in one site, which, albeit sparse, is very large and usually exceeds the available main memory size. The above mentioned approaches however, are not focused on P2P networks, therefore the issue of dynamics is not addressed.

In [29], Sankaralingam et al. presented a P2P algorithm in which the PageRank computation is performed at the network level, with peers constantly updating the scores of their local pages and sending these updated values through the network. Shi et al. [30] also compute PR at the network level, but they reduce the communication among peers by distributing the pages among the peers according to some load-sharing function.

Counting the number of distinct elements in a multiset has been a well studied problem, in particular in the context of database systems [10,15,12]. The recent work by Ntarmos et al [23] considers hash sketched based counting of distinct items leveraging a distributed hash table (DHT). Our own prior work [4] considers the estimation of global document frequency in a P2P Web search engine layered on top of which is DHT. In the area of unstructured networks, which we consider in our current work, there have been a lot of research on the so called gossiping algorithms [16,3,18]. The main idea is to let peers perform random meetings as a background process along with the actual application. Peers constantly sent values to the others peers and updated according to the messages received from the other participants. These approaches are in particular well suited for the application in $\mathrm{J}_{\mathrm{P}}^{\mathrm{X}}$, since $\mathrm{J}_{\mathrm{P}}^{\mathrm{X}}$ relies anyway on random peer meetings. In this paper, we consider a gossip based algorithm based on hash sketches [15] to estimate the number of distinct documents in the system. This differs from standard gossip-based aggregation approaches, which usually focus on computing values such as *max*, *avg*, and *count*.

## 3 JXP Basics

$\mathrm{J}_{\mathrm{P}}^{\mathrm{X}}$ [26] is an algorithm to compute global authority scores in a decentralized manner. In [26] the authors give a mathematical proof that the $\mathrm{J}_{\mathrm{P}}^{\mathrm{X}}$ scores converge to the global

PageRank authority scores, i.e., the scores that would be obtained by a PageRank computation on a hypothetically centralized combined Web graph over all peers.

Running at each peer, $J_P^X$ combines standard PageRank computations on the local portion of the Web graph with condensed knowledge on the rest of the network, which is continuously being refined by meetings with other peers. The knowledge about the non-local partition of the Web graph is collapsed into a *single* dedicated node that is added to the local Web graph, the so-called *world node*. It conceptually represents all non-local documents of the Web graph.

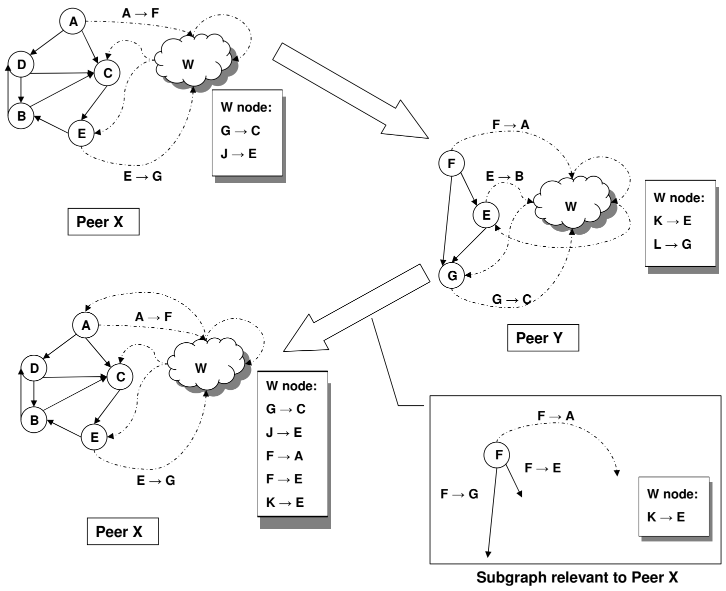

**Fig. 1.** Exchanging local knowledge

This is an application of the state-lumping techniques used in the analysis of large Markov models. All documents of the local Web graph that point to non-local documents will create an edge to the world node (cf. Figure 1).

Meetings with other peers in the network are used to exchange local knowledge and to improve the local approximation of global authority scores, as illustrated in Figure 1. As a peer learns about non-local documents pointing at a local document, a corresponding edge from the world node to that local document is inserted into the local Web graph[1]. Each peer locally maintains a list of scores for external documents that point to a local document. The weight of an edge from the world node to a local document reflects the authority score mass that is transferred from the non-local document; if this edge already exists, its weight is updated with the maximum of both scores. The world node contains an additional self-loop link, representing links within non-local pages. The $J_P^X$ authority score of the world node itself reflects the $J_P^X$ score mass of

[1] Note that such a meeting does *not* increase the number of nodes of a peer's local Web graph.

all non-local pages. Locally, each peer recomputes its local $\mathrm{J}_{\mathrm{P}}^{\mathrm{X}}$ scores by a standard PageRank power iteration on the local Web graph augmented by the world node.

The $\mathrm{J}_{\mathrm{P}}^{\mathrm{X}}$ algorithm is scalable, as the PageRank power iteration computation is always performed on small local graphs, regardless of the number of peers in the network. The local storage requirements at each peer are independent from the number of remote peers they have previously met and the size of the remote (or even the complete) Web graph, i.e., the size of the local Web graph *only* reflects the local crawl. The autonomy of peers is fully preserved by the asynchronous nature of communication and computation.

Current limitations of the algorithm is that it assumes that (i) the global size of the graph is known, and (ii) peers and their contents are static through all the computation. In [25] we addressed (i), showing that a wrong estimation of global size causes only a rescaling of the $\mathrm{J}_{\mathrm{P}}^{\mathrm{X}}$ scores, while the ranking order is preserved. For convergence to the true PageRank scores however, the correct graph size is needed. In case of peer dynamics only, i.e., the Web graph is fixed and peers are constantly leaving and eventually joining the network again, the convergence guarantees given in [26] still hold, with the difference that the convergence is slowed down, given that some peers are not accessible for a certain period. Dealing with content dynamics, i.e., documents being added to the network or becoming unavailable, gives a more realistic model, and is the main contribution of this current work. More details can be found in Section 5.

## 4 Estimating the Global Number Documents

As mentioned earlier, convergence to the true PageRank values requires the knowledge of the total number of documents in the network. In this section we propose a method for computing such value in a dynamic P2P network.

Instead of a single value, peers initialize a hash sketch [15] that represents the set of local pages. During a meeting, peers exchange the hash sketches and the local copy is updated by taking the union of both sketches (local and from the peer met). What we seek is to have the hash sketches at all peers to be the same and equal to the sketch that represents the union of all local sets. The size of the network, can then be estimated at each peer, with error bounds given by the original hash sketch work, which we will briefly discuss in the following section. Thereafter, we discuss how this gossiping algorithm can be applied to dynamic settings using a sliding window approach.

### 4.1 Hash Sketches

Hash sketches were first proposed by Flajolet and Martin in [15] to probabilistically estimate the cardinality of a multi set $S$. Hash sketches rely on the existence of a pseudo-uniform hash function $h() : S \to [0, 1, \ldots, 2^L)$. Durand and Flajolet presented a similar algorithm in [12] (*super-LogLog counting*) which reduced the space complexity and relaxed the required statistical properties of the hash function.

Briefly, hash sketches work as follows. Let $\rho(y) : [0, 2^L) \rightarrow [0, L)$ be the position of the least significant (leftmost) 1-bit in the binary representation of $y$; that is,

$$\rho(y) = \min_{k \geq 0} bit(y, k) \neq 0, \ y > 0$$

and $\rho(0) = L$. $bit(y, k)$ denotes the $k$-th bit in the binary representation of $y$ (bit-position $0$ corresponds to the least significant bit). In order to estimate the number $n$ of distinct elements in a multi set $S$ we apply $\rho(h(d))$ to all $d \in S$ and record the least-significant 1-bits in a bitmap vector $B[0 \ldots L-1]$. Since $h()$ distributes values uniformly over $[0, 2^L)$, it follows that

$$P(\rho(h(d)) = k) = 2^{-k-1}$$

Thus, when counting elements in an $n$-item multi set, $B[0]$ will be set to 1 approximately $\frac{n}{2}$ times, $B[1]$ approximately $\frac{n}{4}$ times, etc. Then, the quantity $R(S) = max_{d \in S} \rho(d)$ provides an estimation of the value of $\log_2 n$. The authors in [15,12] present analysis and techniques to bound from above the error introduced.Techniques which provably reduce the statistical estimation error typically rely on employing multiple bitmaps for each hash sketch, instead of only one. The overall estimation then is an averaging over the individual estimations produced using each bitmap.

Hash sketches offer duplicate elimination “for free”, or in other words, they allow counting distinct elements in multi sets. Estimating the number of distinct elements (e.g., documents) of the *union* of an arbitrary number of multi sets (e.g., distributed and autonomous collections) - each represented by a hash sketch synopsis - is easy by design: a simple bit-wise $OR$-operation over all synopses yields a hash sketch for the combined collection that instantly allows us to estimate the number of distinct documents of the combined collection.

## 4.2 Estimating Global Counts Using Hash Sketches

In the task of estimating global counts using hash sketches, peers can benefit from the counts of all the other peers, due to the duplicate aware counting. To make the analysis tractable, lets assume for now that all peers perform their meetings in a synchronized way, i.e. after some amount of time, all peers have performed the same number of meetings. Consider one particular peer that is about to perform its $m^{th}$ meeting. Obviously, it has already performed $m-1$ meetings in the past, as well as the peer it will meet in its $m^{th}$ meeting. In total, both peers now double the amount of meetings they are aware of (recorded in hash sketches). We denote $C(m)$ the number of meetings a peer is aware of after the $m^{th}$ meeting. It is easy to see that we can also write $C(m) = 2^{(m-1)}$, i.e., the number of meetings a peer is aware of grows exponentially with the number of meetings the peer has performed. From a single peer's point of view, after having performed $m$ meetings the situation is identical with having had $C(m)$ meetings where peers do not share information about their previous meetings.

Charikar et al [10] consider the problem of estimating the number of distinct values in a column of a table. The difference to our scenario is that in a database table, the number of tuples is known, whereas in a truly distributed large scale system, the total number of peers is unknown. In addition, we know only how many peers or documents we have seen so far, and not the frequency of observation. In practice, all we have is an estimate of distinct values given the sampling using meetings and the exchanged hash sketches, thus we cannot directly apply the estimators from [10]. The estimation of the number of distinct items, however, in a multi set is a well studied problem (cf., e.g., [21]). We will now briefly discuss on how many distinct meetings will in expectation be among the $C(m)$ meetings.

Applying the results from [21], given $C(m)$ samples of a multi set that contains $n$ distinct elements, the expected number of elements is

$$E[distinctItems] = n(1 - e^{-C(m)/n})$$

According to [21] it can be derived that

$$\frac{C(m)}{n} = ln(\frac{n}{n - E[distinctItems]})$$

and this expression can be used to get an estimator $\overset{\wedge}{n}$ of the total number of distinct elements $n$. Then, the variance of the estimator is given by

$$\sigma^2_{\overset{\wedge}{n}} = \frac{n}{e^{C(m)/n} - (1 + \frac{C(m)}{n})}$$

Hence, to reach an negligible error even for really huge $n$, we need only few rounds of peer meeting since $C(m)$ grows exponentially.

How can we now estimate the number of documents we did not see during the estimation process. Note that this is not the same as the error in estimating the number peers (distinct meetings) since the data is not evenly spread across the peers with some peers being small and some peers being extremely large. Due to the fast estimation process, the case that few peers are not observed after some time by the majority of peers becomes very unlikely.

In practice we do not know the value of $C(m)$ since peers meet asynchronously and the online time of peers largely varies. In addition, we are of course not aware of $n$, the total distinct number of elements (peers or documents) in the system. We have only an estimate given by the hash sketch based sampling. The reasoning presented above shows, however, that few iterations are needed to get to a meaningful hash sketch. That does of course not include any reasoning about the quality of the estimated obtained by the hash sketches which is given in the original work by Flajolet et al and is thus nicely orthogonal to our goals.

The number of documents changes here as well as in the case when documents are leaving the system, numbers can increase or decrease. The former case can be easily handled by the estimator introduced above. The latter case requires some further improvements. Since one can easily add items to a hash sketch but one cannot remove items from such a sketch, we employ the usage of a time sliding window over multiple hash sketches. We let each peer keep an array of $k$ hash sketches, ordered by time, the $k^{th}$ hash sketch is considered to be the “oldest” one. After $\tau$ time steps we remove the oldest sketch and insert an empty one at array position 1. At any time, the current estimate of distinct items is the estimate derived from the hash sketch created by forming the union of all $k$ sketches. Obviously, newly observed items will be inserted into the sketch at position 1.

## 5 Handling Dynamics

Recalling the previous $\mathrm{J}_{\mathrm{P}}^{\mathrm{X}}$ meeting procedure, a peer selects another peer for a meeting and contacts this peer. The contacted peer then returns the information that is relevant to the peer initiator. This information consists of a list of all external pages known to the contacted peer (local + world node) that contain links to local pages at the peer initiator. Due to possible overlaps and the asynchronous nature of the algorithm, different peers might provide different score values for the same page. In these cases, the highest score is kept, since the correctness proof of the algorithm shows that scores are, at any time during the computation, upper-bounded by the true PageRank scores, i.e., the scores to which the $\mathrm{J}_{\mathrm{P}}^{\mathrm{X}}$ scores converge to. Therefore, keeping the highest values provides a speed up in convergence. In addition, local pages with links to pages outside the local graph do not need to know the exact location of those, since links to non-local documents are represented as links to the world node. With dynamics, however, three new events come into play, and the algorithm needs to detect them: pages can be added, modified, or deleted.

### 5.1 The New World Node

So far we actually did not consider the problem of invalid information kept in the World node in case of peers and/or documents leaving the system. One idea would be to keep for each document in the world node that points to a local page a list of peers that had reported a score for that particular document. The number of data to keep track at (bookkeeping) should be constant or growing sub linearly, e.g. in the order of $log(N)$ like in the case of $N$ peers in a *Distributed Hash Tables* (DHTs) [1,31,27,28], where $N$ is the number of peers in the network. Keeping track of all peers that store a particular document is out of question, since it would require massive amount of storage caused by overly popular pages, like for instance, `google.com` or `cnn.com`.

Instead of using a sketch based representation to keep track about the peers that have reported scores for a particular page, we opt for storing the last $\chi$ peers that reported a score, i.e., we store for each document a list of pairs (peerId, score) for the last $\chi$ scores seen for the page, along with the corresponding peer. The parameter $\chi$ can depend on the storage capacity of each peer, but we envision $\chi$ to be in order of $O(log\ N)$. This limitation to a certain length is reasonable, since the probability that all peers inside a per-document list leave is small, (analog to the size of “finger tables” in DHTs). Even if a peer removes a particular page and that page is actually in the system, it will be rediscovered, due to the basic $\mathrm{J}_{\mathrm{P}}^{\mathrm{X}}$ performance. Hence, the actual choice of $\chi$ is not overly crucial for the performance of $\mathrm{J}_{\mathrm{P}}^{\mathrm{X}}$.

In addition to remembering external pages with outgoing edges to the local graph, the world node now needs also to keep track of external pages that are *pointed by* local pages. This way we can correctly reconstruct both links from and to the world node. Here we also apply the approach of keeping a list of limited size containing the last $\chi$ peers met that contained the page, but no score is needed, since they do not directly influence the local scores.

## 5.2 JXP Meetings Adapted

In the previous version of $J_P^X$, the meetings are of fundamental importance for the effectiveness and correctness of the algorithm. With dynamics, its role becomes even more crucial: it is through the meetings that peers will be able to detect the changes in the network. As stated before, a change can be of one of the three types: pages can be added, modified, or deleted.

Page addition is a trivial problem, since the algorithm is already designed to discover non-local documents. For speeding up convergence, a peer also sends information about pages currently in its world node to the meeting initiator (like in the previous version) . With scores lists instead of single scores, a decision has to be made about what to send for those pages. For keeping message cost small, our solution is to send a single (peerId, score) pair per page, where the score is computed by averaging all scores current known for the page. With the limit on the size of the lists, and a fair amount of meetings performed, old scores will gradually be replaced by updated, better scores, and the average is then expected to converge to the correct score of the page. For the peerId, we can simply choose the most recent peer met for that page, since chances are higher that this peer will remain for a longer period in the network.

Page deletion might occur when peers that reported information for the page have left the network or changed their contents. Whenever one of the two happens, the reference for that peer is removed from the world node. If the list of peers for a document becomes empty, it is assumed that the page no longer exist, and therefore must be removed from the world node.

It could also happen that a page had its contents modified, so it could still be reached but the new information given for that page contradicts previous information. By comparing what is already stored on the world node against what is being reported by the other peer we can detect such events. Changes on the score are not considered, since peers are constantly updating this information. What is checked is whether the outgoing edges have been modified. If so, the page is initially removed from the world node and re-added with the new information.

What is left to describe is how to detect when a peer has left the system. In P2P networks, it is very common that peers temporally leave the network and return to it a short later. In such situations, we would rather leave the world node unchanged and wait until the peer returns. Therefore, a single failed attempt to contact a peer sometimes might not be an good indication that the peer has left the network indefinitely. Instead, we keep a counter of number of consecutive failed attempts made to contact a peer, and only if this number is above a certain threshold, that can be tuned according to the network behavior, we declare that the peer no longer alive, and its references should be removed. During a successful attempt this counter is reset.

## 6 Experiments

### 6.1 Experimental Setup

$\mathrm{J}_\mathrm{P}^\mathrm{X}$ peers are implemented in Java 1.6, the peers' data collections (i.e., their local graphs) obtained by performing independent crawls on the *eu-2005* dataset, available under `http://law.dsi.unimi.it/`, and accessible using the WebGraph framework [6], available under `http://webgraph.dsi.unimi.it/`. The dataset was obtained in 2005 by crawling parts of the .eu domain, and contains 862,664 documents with 19,235,140 links. For a meeting, a peer contacts a randomly chosen peer in the network, and asks for its current local knowledge.

For evaluating the performance we compare the authority scores given by the $\mathrm{J}_\mathrm{P}^\mathrm{X}$ algorithm against the true PageRank scores of pages in the complete collection. Since, in the $\mathrm{J}_\mathrm{P}^\mathrm{X}$ approach, the pages are distributed among the peers and for the true PageRank computation the complete graph is needed, in order to compare the two approaches we construct a total ranking from the distributed scores by essentially merging the score lists from all peers. Since we are trying to evaluate the performance of $\mathrm{J}_\mathrm{P}^\mathrm{X}$ under network churn, the evaluation becomes more complicated, since the baseline, i.e., the PageRank scores of all pages currently available in the system, is not static anymore. Hence, for every change in the network, we consider the union of all pages currently in the system, and compute the baseline scores. For each of these points, we let each active peer also report its current view of the global state. We ignore at runtime how well $\mathrm{J}_\mathrm{P}^\mathrm{X}$ performs and let peers run completely independent, however, each peer reports after each meeting its current view on the global state. After the run we reconstruct at any point back in time what documents have been indexed. Then for some points in time we run the PageRank method for these documents and compare with what peers reported.

The total top-k ranking given by the $\mathrm{J}_\mathrm{P}^\mathrm{X}$ algorithm and the ranking given by traditional, centralized PageRank are compared using the scaled footrule distance [14,13], the weights contributions of elements based on the size of the rankings they are present in. More formally, given the local ranking $\tau$ and the global ranking $\sigma$, $F(\sigma,\tau) = \sum_{i\in\tau} |\sigma(i)/|\sigma| - \tau(i)/|\tau||$, where $\sigma(i)$ and $\tau(i)$ are the positions of the page $i$ in the respective rankings. The measure is normalized by dividing it by $|\tau|/2$. We also use a *linear score error* measure, which is defined as the average of the absolute difference between the $\mathrm{J}_\mathrm{P}^\mathrm{X}$ score and the global PR score over the top-k pages in the centralized PR ranking. In addition, we report on the cosine similarity between the two vectors and the L1-norm of the vector containing the $\mathrm{J}_\mathrm{P}^\mathrm{X}$ scores.

To model peer behavior, we use previous works [22,24] that have derived mathematical models that closely represent the dynamics observed in P2P networks. More specifically, peer joins are expected to follow a Poisson distribution, i.e., the probability that $n$ peers join the network on the next time interval can be written as $P_\lambda(n) = \frac{\lambda^n}{n!}e^{-\lambda}$, where $\lambda$ is the average number of peers joining the network per time interval. Peer leaves, in turn, follow an exponential distribution. Given the average number of drop outs in one time interval ($\mu$), the probability that a peer leaves the network after $x$ time intervals is $F(x) = 1 - e^{-\mu x}$. Note that the interval between two consecutive Poisson events also follows an exponential distribution. In the following experiments, we used

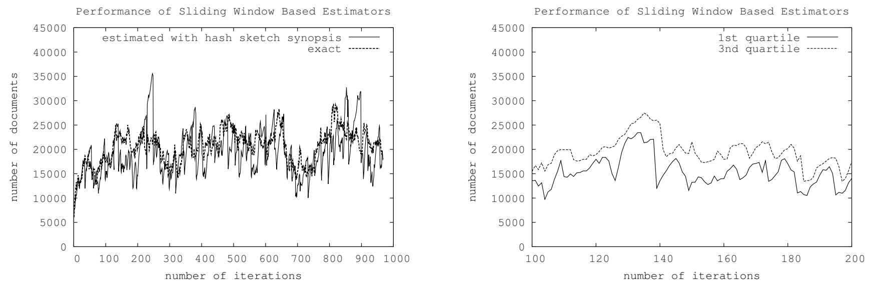

**Fig. 2.** Hash sketch based estimation of the number of documents under network churn

these models to generate peer dynamics. For the content dynamics, we randomly choose a percentage of the peers and replace their local graphs by performing new crawls.

## 6.2 Experimental Results

The experimental evaluation consists of two parts. First we report on the performance of the estimator presented in Section 4. Second, we present results on the performance of $\mathrm{J}_{\mathrm{P}}^{\mathrm{X}}$ under dynamics, which is the main focus of this paper.

Figure 2 (left) shows a quality of the document estimator compared to the exact values, i.e., the number of documents currently in the system. For this experiment, we simulated random peer meetings within a system of 50 peers. Each peer randomly draws from a pool of $150,000$ documents between $250$ and $1000$ distinct documents. Peers are either active or inactive, according to the above mentioned exponential distributions that models the peer behavior. Each peer maintains only $4$ hash sketches with $2^{10}$ bitmaps each, resulting in a negligible storage consumption of $32KByte$. After 2 meetings, each peer shifts the sliding window over the hash sketches by one position, i.e., each hash sketch is valid only for 2 meetings. As shown in Figure 2 (left), the estimation accurately follows the exact values, with major drastic fluctuations being smoothed out. To get a deeper insight about the usability of our estimator inside $\mathrm{J}_{\mathrm{P}}^{\mathrm{X}}$, we also report on the distribution of count estimates, as presented in Figure 2 (right). The variation between the first and the third quartile is remarkably small, indicating that peers nearly agree on one particular value, which is important for the performance of $\mathrm{J}_{\mathrm{P}}^{\mathrm{X}}$. Note that both figures shows one particular, representative run, and that it is not smoothed over multiple runs or multiple parameter choices.

For the experiments with the adapted $\mathrm{J}_{\mathrm{P}}^{\mathrm{X}}$ we increased the size of the network to $1000$ peers. Overlaps among local graphs are allowed, and the collection of all peers holds in total around $100,000$ documents. Peer and content dynamics are introduced in the system always after a certain number of meetings has occurred in the network. We considered both successful and unsuccessful meetings for the counter. We then varied the parameters of the peer churn and content dynamics models, to simulate different degrees of dynamics.

We present results for two scenarios: *Moderate Churn*, with join and leave rates of $100/0.1$, and a change of the contents of $1\%$ of peers; and *Heavy Churn*, with join and

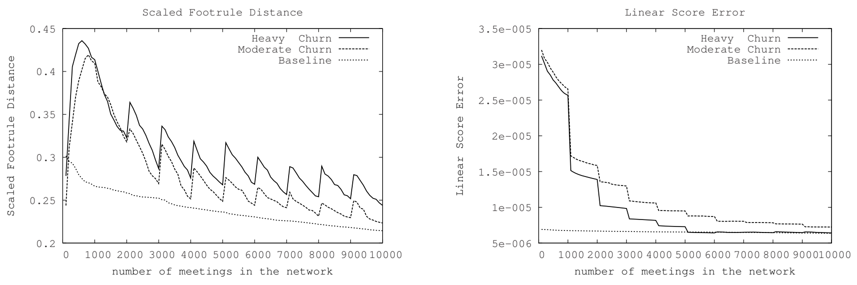

**Fig. 3.** Scaled Footrule distance (left) and Linear Score Error (right)

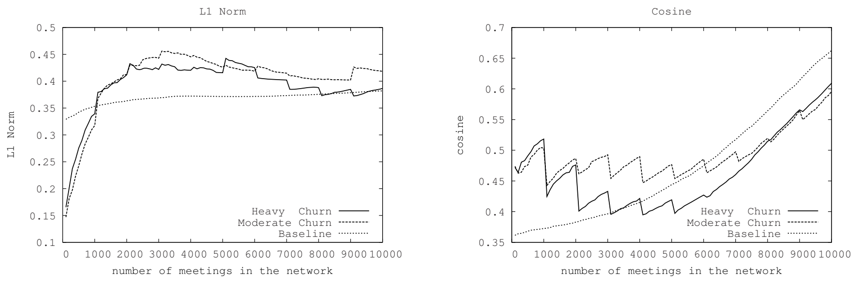

**Fig. 4.** L1-norm (left) and Cosine (right)

leave rates of $200/0.1$, and a change of the contents of $5\%$ of peers. For a better understanding of the impact of dynamics the following results were obtaining without the use of our document estimator, and peers were artificially told about the correct size of the network. Figures 3 and 4 show the results obtained, where the baseline simulates the case where there is no dynamics. Note that the actual values of the linear score error are in general not meaningful: since scores correspond to stationary probability, they are expected to sum up to one, so if there is an increase of the number of pages in the network, the scores drop, which explain the behavior of the curve. However, the key insight obtained here is that the error decreases even under dynamics. The other three accuracy measures show a very nice performance of $\mathrm{J}^{\mathrm{X}}_{\mathrm{P}}$ under churn, in particular the L1-norm nicely follows the baseline, even though the underlying network (data) is not stable.

## 7 Conclusion

In this paper we have addressed the problem of computing distributed PageRank authority scores with particular emphasis on the key requirements to address the challenges of highly dynamic systems. We have identified the main shortcomings of our $\mathrm{J}^{\mathrm{X}}_{\mathrm{P}}$ method, and presented means to extend the algorithm to cope with network dynamics. We have presented an estimator based on hash sketches and sliding windows to count the number of distinct documents in a dynamic network, which is one of the basic input parameters of $\mathrm{J}^{\mathrm{X}}_{\mathrm{P}}$. Secondly, but perhaps even more importantly, we have presented several modifications to the original $J_P^X$ data structures that allow for accurate PageRank computation even under high churn, while keeping storage requirements and message costs low. As future work we can think of ways of automatically detecting the dynamic behavior that would allow peers to adjust their local settings, for an even better performance.

## References

1. Aberer, K.: P-grid: A self-organizing access structure for p2p information systems. In: Batini, C., Giunchiglia, F., Giorgini, P., Mecella, M. (eds.) CoopIS 2001. LNCS, vol. 2172, pp. 179–194. Springer, Heidelberg (2001)
2. Abiteboul, S., Preda, M., Cobena, G.: Adaptive on-line page importance computation. In: WWW Conference, pp. 280–290. ACM Press, New York (2003)
3. Bawa, M., Gionis, A., Garcia-Molina, H., Motwani, R.: The price of validity in dynamic networks. J. Comput. Syst. Sci. 73(3), 245–264 (2007)
4. Bender, M., Michel, S., Triantafillou, P., Weikum, G.: Global document frequency estimation in peer-to-peer web search. In: WebDB (2006)
5. Berkhin, P.: A survey on pagerank computing. Internet Mathematics 2(1), 73–120 (2005)
6. Boldi, P., Vigna, S.: The webgraph framework i: compression techniques. In: WWW, pp. 595–602 (2004)
7. Borodin, A., Roberts, G.O., Rosenthal, J.S., Tsaparas, P.: Link analysis ranking: algorithms, theory, and experiments. ACM TOIT 5(1), 231–297 (2005)
8. Brin, S., Page, L.: The anatomy of a large-scale hypertextual web search engine. In: WWW7, pp. 107–117 (1998)
9. Chakrabarti, S.: Mining the Web: Discovering Knowledge from Hypertext Data. Morgan-Kauffman, San Francisco (2002)
10. Charikar, M., Chaudhuri, S., Motwani, R., Narasayya, V.R.: Towards estimation error guarantees for distinct values. In: PODS, pp. 268–279 (2000)
11. Chien, S., Dwork, C., Kumar, R., Simon, D.R., Sivakumar, D.: Link evolution: Analysis and algorithm. Internet Mathematics 1(3), 277–304 (2004)
12. Durand, M., Flajolet, P.: Loglog counting of large cardinalities (extended abstract). In: Di Battista, G., Zwick, U. (eds.) ESA 2003. LNCS, vol. 2832, pp. 605–617. Springer, Heidelberg (2003)
13. Dwork, C., Kumar, S.R., Naor, M., Sivakumar, D.: Rank aggregation methods for the web. In: WWW, pp. 613–622 (2001)
14. Fagin, R., Kumar, R., Sivakumar, D.: Comparing top k lists. In: SIAM Discrete Algorithms (2003)
15. Flajolet, P., Martin, G.N.: Probabilistic counting algorithms for data base applications. J. Comput. Syst. Sci. 31(2), 182–209 (1985)
16. Jelasity, M., Montresor, A., Babaoglu, Ö.: Gossip-based aggregation in large dynamic networks. ACM Trans. Comput. Syst. 23(3), 219–252 (2005)
17. Kamvar, S., Haveliwala, T., Manning, C., Golub, G.: Exploiting the block structure of the web for computing pagerank. Technical report, Stanford University (2003)
18. Kempe, D., Dobra, A., Gehrke, J.: Gossip-based computation of aggregate information. In: FOCS, Washington, DC, USA, p. 482. IEEE Computer Society, Los Alamitos (2003)
19. Langville, A., Meyer, C.: Updating the stationary vector of an irreducible markov chain with an eye on google's pagerank. In: SIMAX (2005)
20. Langville, A.N., Meyer, C.D.: Deeper inside pagerank. Internet Mathematics 1(3), 335–400 (2004)

21. Lewontin, R., Prout, T.: Estimation of the number of different classes in a population. Biometrics 12(2), 211–233 (1956)
22. Liben-Nowell, D., Balakrishnan, H., Karger, D.R.: Analysis of the evolution of peer-to-peer systems. In: PODC, pp. 233–242 (2002)
23. Ntarmos, N., Triantafillou, P., Weikum, G.: Counting at large: Efficient cardinality estimation in internet-scale data networks. In: ICDE, p. 40 (2006)
24. Pandurangan, G., Raghavan, P., Upfal, E.: Building low-diameter p2p networks. In: FOCS, pp. 492–499 (2001)
25. Parreira, J.X., Castillo, C., Donato, D., Michel, S., Weikum, G.: The juxtaposed approximate pagerank method for robust pagerank approximation in a peer-to-peer web search network. VLDB J. 17(2), 291–313 (2008)
26. Parreira, J.X., Donato, D., Michel, S., Weikum, G.: Efficient and decentralized pagerank approximation in a peer-to-peer web search network. In: VLDB, pp. 415–426 (2006)
27. Ratnasamy, S., Francis, P., Handley, M., Karp, R.M., Shenker, S.: A scalable content-addressable network. In: SIGCOMM, pp. 161–172 (2001)
28. Rowstron, A.I.T., Druschel, P.: Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In: IFIP/ACM Middleware, pp. 329–350 (2001)
29. Sankaralingam, K., Yalamanchi, M., Sethumadhavan, S., Browne, J.C.: Pagerank computation and keyword search on distributed systems and p2p networks. J. Grid Comput. 1(3), 291–307 (2003)
30. Shi, S., Yu, J., Yang, G., Wang, D.: Distributed page ranking in structured p2p networks. In: ICPP (2003)
31. Stoica, I., Morris, R., Karger, D., Kaashoek, M.F., Balakrishnan, H.: Chord: A scalable peer-to-peer lookup service for internet applications. In: SIGCOMM, NY, USA, pp. 149–160. ACM Press, New York (2001)
32. Wang, Y., DeWitt, D.J.: Computing pagerank in a distributed internet search system. In: VLDB (2004)
33. Wu, J., Aberer, K.: Using a Layered Markov Model for Distributed Web Ranking Computation. In: ICDCS (2005)