Efficient and Decentralized PageRank Approximation in a Peer-to-Peer Web Search Network [*]

Josiane Xavier Parreira[†], Debora Donato[‡], Sebastian Michel[†], Gerhard Weikum[†]

[†] Max-Planck Institute for Computer Science
66123 Saarbrücken, Germany
{jparreir,smichel,weikum}@mpi-inf.mpg.de

[‡] Universita di Roma "La Sapienza"
00198 Roma, Italy
donato@dis.uniroma1.it

ABSTRACT

PageRank-style (PR) link analyses are a cornerstone of Web search engines and Web mining, but they are computationally expensive. Recently, various techniques have been proposed for speeding up these analyses by distributing the link graph among multiple sites. However, none of these advanced methods is suitable for a fully decentralized PR computation in a peer-to-peer (P2P) network with autonomous peers, where each peer can independently crawl Web fragments according to the user's thematic interests. In such a setting the graph fragments that different peers have locally available or know about may arbitrarily overlap among peers, creating additional complexity for the PR computation.

This paper presents the JXP algorithm for dynamically and collaboratively computing PR scores of Web pages that are arbitrarily distributed in a P2P network. The algorithm runs at every peer, and it works by combining locally computed PR scores with random meetings among the peers in the network. It is scalable as the number of peers on the network grows, and experiments as well as theoretical arguments show that JXP scores converge to the true PR scores that one would obtain by a centralized computation.

1. INTRODUCTION

One of the cornerstones of Web search engines and Web mining is link analysis for authority scoring, most notably, the two seminal methods PageRank (PR) by Brin and Page [8] and HITS by Kleinberg [23]. Both methods are Eigenvector-based algorithms that determine the importance of a page based on the importance of the pages that point to it. Their computation is quite expensive as it involves iteratively computing the principal Eigenvector of a matrix derived from the Web link graph. An alternative but equivalent view of PR is that it computes stationary probabilities of a Markov chain that corresponds to a random walk on the Web. Recent work has made progress on efficiently computing PR-style authority scores [20, 11, 14, 27], but the high storage demand of the – sparse but nonetheless huge – underlying matrix seems to limit this kind of link analysis to a central server with very large memory.

Recently, various techniques have been proposed for speeding up these analyses by distributing the link graph among multiple sites [21, 40, 2]. In fact, given that Web data is originally distributed across many owner sites, it seems a much more natural (but obviously also more challenging) computational model to perform parts of the PR computation right where the data originates from followed by smaller distributed computation for combining the local results in an appropriate way. Exploiting a block structure in the link matrix is an example [21]. However, these advanced methods work only when the overall Web graph is nicely partitioned into disjoint fragments, which is the case when partitions are formed by the sites that own the pages.

A different distributed architecture that has gained significant momentum is peer-to-peer (P2P) systems. P2P technology is a compelling paradigm for large-scale file sharing, publish-subscribe, and collaborative work, as it provides great scalability and robustness to failures and very high dynamics (so-called churn) [1, 38, 32, 33]. Another intriguing P2P application could be Web search: spreading the functionality and data of a search engine across thousands or millions of peers. Such an architecture is being pursued in a number of research projects [39, 31, 4] and could offer various key advantages: lighter load and smaller data volume per peer, and thus more computational resources per query and data unit, could enable more powerful linguistic or statistical learning methods; with each peer being close to the human user and the user trusting its local software and controlling the degree of sharing personal information and collaboration with other peers, there is a great opportunity for leveraging user behavior such as explicit or implicit feedback in the form of query logs, click streams, or bookmarks; and finally, a decentralized approach could provide better immunity to search result distortion by the bias of big providers, commercial interests, or even censorship.

In this paper we consider the architecture of a P2P search engine where each peer is autonomous, crawls Web fragments and indexes them locally according to the user's interest profile, and collaborates with other peers for query routing and execution. Queries would often be executed locally on the user's personalized "power search engine", and occasionally forwarded to other peers for better results. In such a setting, PR-style scores are still crucial for the ranking of search results, but the local Web fragment of a peer may be too small or incomplete for a meaningful link anal-

[*]Partially supported by the EU within the 6th Framework Programme under contract 001907 "Dynamically Evolving, Large Scale Information Systems" (DELIS).

ysis. Distributed PR computations of the kind mentioned above seem natural, but they work only for disjointly partitioned graphs; in our setting we face the additional complexity posed by the fact that the graph fragments of different peers may arbitrarily overlap.

JXP (Juxtaposed Approximate PageRank) is an algorithm for coping with this situation: dynamically computing, in a decentralized P2P manner, global authority scores when the Web graph is spread across many autonomous peers with arbitrary overlapping and the peers are a priori unaware of other peers' fragments. The ideas for JXP have appeared in a preliminary short paper at a workshop [30]. The current paper elaborates these ideas, provides mathematical underpinnings, including a convergence proof (which were missing in the workshop paper), and develops novel extensions and run-time enhancements, along with more comprehensive experimental studies.

In the JXP algorithm, each peer computes the authority scores of the pages that it has in its local index, by locally running the standard PR algorithm. A peer gradually increases its knowledge about the rest of the network by meeting with other, randomly chosen, peers and exchanging information, and then recomputing the PR scores of local interest. This process, in principle, runs forever, and experiments have indicated that the resulting JXP scores quickly converge to the true, global PR scores.

For further improving the network performance, we propose a heuristic strategy for guiding the choice of peers for a meeting. The improvements can be observed in our experimental results with real-world data collections. We provide an mathematical framework for the analysis of JXP, where some important properties are highlighted and the proof that the JXP scores converge to the true global PR scores is given. An application of the algorithm is also given, where we have integrated the JXP scores into a P2P search engine in order to improve the ranking of the results.

The rest of the document is organized as follows. Section 2 discusses related work. A more detailed explanation of the JXP algorithm is given in Section 3. The extensions and run-time improvement of JXP are discussed at Section 4, and the mathematical analysis is given at Section 5. Experimental results are described in Section 6, and Section 7 presents ideas for future work.

2. RELATED WORK

Link-based authority ranking has received great attention in the literature. It has started with the seminal works of Brin and Page [8] and Kleinberg [23], and after these, many other models and techniques have followed. Good surveys of the many improvements and variations are given in [12, 26, 7, 5].

2.1 PageRank

The basic idea of PR 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 this Markov chain (i.e., the existence of stationary page-visit probabilities), additional random jumps to uniformly chosen target pages are allowed with small probability $(1-\epsilon)$. Formally, the PR of a page q is defined as:

$$PR(q) = \epsilon \times \sum_{p|p\rightarrow q} PR(p)/out(p) + (1-\epsilon) \times 1/N$$

where N is the total number of pages in the link graph, $PR(p)$ is the PR score of the page p, $out(p)$ is the outdegree of p, the sum ranges over all link predecessors of q, and $(1-\epsilon)$ is the random jump probability, with $0 < \epsilon < 1$ and usually set to a value like 0.85.

PR values are usually computed by initializing a PR vector with uniform values $1/N$, and then applying a power iteration method, with the previous iteration's values substituted in the right-hand side of the above equation for evaluating the left-hand side. This iteration step is repeated until sufficient convergence, i.e., until the PR scores of the high-authority pages of interest exhibit only minor changes.

2.2 Distributed PageRank

With the advent of P2P networks [1, 38, 32, 33] attention to distributed link analysis techniques has been growing.

In [40] Wang and DeWitt presented a distributed search engine framework, in which the authority score of each page is computed by performing the PR 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 PR values by combining local page authority scores and server authority values. Wu and Aberer [41] 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. [21] that postulates a block structure of the link matrix and exploits this structure for faster convergence of the global PR computation. A drawback from these approaches is the need of a particular distribution of pages among the sites, where the graph fragments have to be disjoint — a strong constraint, given that in most P2P networks peers are completely autonomous and crawl and index Web data at their discretion, resulting in arbitrarily overlapping graph fragments.

Chen et al. [13] proposed a way of approximating the PR value of a page locally, by expanding a small subgraph around the page of interest, placing an estimated PR at the boundary nodes of the subgraph, and running the standard algorithm. This approach assumes that the full link structure is accessible at a dedicated graph server. In a P2P scenario, however, this algorithm would require the peers to query other peers about pages that have links to their local nodes, and pages that point to pages that point to local pages, and so on. This would be a significant burden for a highly dynamic P2P network. The JXP algorithm, on the other hand, requires much less interaction among peers, and with the new peer selection strategy, the number of interactions is even smaller.

Other techniques [25, 14] for approximating PR-style authority scores with partial knowledge of the global graph use state-aggregation technique from the stationary analysis of large Markov chains. These techniques have been developed for the purpose of incremental updates to authority scores when only small parts of the graph have changed. Dynamic computation in a P2P network is not an issue in this prior work. Another work related to this topic is the one by Broder and Lempel [11], where they have presented a graph aggregation method in which pages are partitioned into hosts and the stationary distribution is computed in a two-step approach, combining the stationary distribution inside the host and the stationary distribution inter-hosts.

A storage-efficient approach to computing authority scores is the OPIC algorithm developed by Abiteboul et al. [3]. 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. It does so by randomly (or otherwise fairly) visiting Web pages in a long-running crawl process and performing a small step of the PR power iteration (the numeric technique for computing the principal Eigenvector) for the page and its successors upon each such visit. The bookkeeping for tracking the gradually approximated authority of all pages is carried out at a central site, the Web-warehouse server. This is not a P2P algorithm either.

In [34], Sankaralingam et al. presented a P2P algorithm in which the PR 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. [35] 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. In contrast to these P2P-style approaches, JXP algorithm performs the actual computations locally at each peer, and thus needs a much smaller number of messages.

3. THE JXP ALGORITHM

The goal of the JXP algorithm is to approximate global authority scores by performing local computations only, with low storage costs, and a moderate number of interactions among peers. It runs on every peer in the network, where each peer stores only its own local fragment of the global graph. The algorithm does not assume any particular assignment of pages to peers, and overlaps among the graph fragments of the peers are allowed.

The idea of the algorithm is simple, yet it is quite powerful. Starting with the local graph G of a peer, the peer first extends G by adding a special node W, called *world node* since its role is to represent all pages in the network that do not belong to G. An initial JXP score for local pages and the world node is obtained by running the PR algorithm in the extended local graph $G' = G+W$. The results are stored in a score list L. This initialization procedure is described in Algorithm 1.

Algorithm 1 JXP Initialization Procedure

```
1: input: local graph $G$ and est. size of global graph $N$
2: n ← size($G$)
3: Create world node $W$
4: $score(p|p \in G) \leftarrow 1/N$
5: $score(W) \leftarrow (N-n)/N$
6: $G' \leftarrow (G+W)$
7: $PR \leftarrow PageRank(G')$
8: $L \leftarrow PR$
```

JXP assumes that the total number of nodes in the global graph is known or can be estimated with decent accuracy. This is not a critical assumption; there are efficient techniques for distributed counting with duplicate elimination, and JXP could even be modified to work without this estimate, but our presentation here will make use of the assumption.

The world node has special features, regarding its own score and how it is connected to the local graph. As it represents all the pages not indexed by the peer, we take all the links from local pages to external pages and make them point to the world node. In the same way, as the peer learns about external links that point to one of the local pages, we assign these links to the world node. (This is when the peer meets with another peer). For a better approximation of the total authority score mass that is received from external pages, we weigh every link from the world node based on how much of the authority score is received from the original page that owns the link. Another special feature of the world node is that it contains a self-loop link, that represents links from external pages pointing to other external pages. The score of the world node is equivalent to the sum of the scores of the external pages. During the local PR computation the probability of a random jump to the world node is also set proportional to the number of external pages.

Since local information is not sufficient to estimate global scores, peers improve their knowledge by meeting other peers in the network and exchanging the information they currently have, namely the extended local graph and the score list. The information is then combined by both of the two meeting peers, asynchronously and independently of each other. This works as follows. A new graph is formed from the union of both local graphs. World nodes are also unified to create a new world node that is connected to the new graph. The union of two world nodes consists of taking the union of the links that are represented in them and removing those links that already appear at the graph to which the new world node will be attached to, so multiple representations of the same link are avoided.

More formally, let $G_A(V_A, E_A)$ be the local graph at peer A, where V_A and E_A are the sets of pages and links, respectively. Let $W_A(T_A)$ be the world node attached to peer's A local graph, where T_A is the set of links represented at the world node. When peer A exchange information with peer B, they both create locally a merged graph $G_M(V_M, E_M)$, where $V_M = V_A \cup V_B$ and $E_M = E_A \cup E_B$, and a new merged world node $W_M(T_M)$ that it is connected to G_M, where $T_M = (T_A \cup T_B) - E_M$, i.e., the set of links outgoing from pages that are not in V_M with target nodes inside V_M.

A new merged list of scores, L_M, is created by merging the two original lists, taking the average of the scores for the pages that belong to both of them.

After this merging step, the peer performs the PR algorithm on the extended graph $G_M + W_M$, using the scores from L_M as initial scores. The score of the world node is initially set to

$$L_M(W) = 1 - \sum_{i \in V_M} L_M(i) \tag{1}$$

and the PR scores obtained, PR, are used to update the current JXP score list L_M in the following manner:

$$L'_M(i) = \begin{cases} PR(i) & \text{if } i \in V_M \\ \frac{L_M(i) \times PR(W)}{L_M(W)} & \text{otherwise} \end{cases} \tag{2}$$

The next step is to update local score list L_A and local world node W_A. L'_A is derived from L'_M by keeping the scores of all pages that either belong to V_A or point to one of the pages in V_A. W'_A is obtained by taking all the links from W_M that point to a page in V_A and adding the links from E_B that also point to a page in V_A. This is done analogously at peer B.

The merged graph G_M, merged node W_M and merged score list L_M are then discarded, as well as G_B, W_B and L_B, so that the storage requirements are kept low. Algorithm 2 shows the pseudo code of the JXP algorithm. Figure 1 illustrates the procedures to combine and disconnect local graphs and world nodes.

4. EXTENSIONS AND OPTIMIZATIONS

The JXP algorithm, as presented before, already has nice

Algorithm 2 The JXP Algorithm

```
1: **input:** local graph $G_A$, world node $W_A$, score list $L_A$
2: **repeat**
3:     Contact a random peer $B$ in the network and exchange information
4:     $G_M \leftarrow mergeGraphs(G_A, G_B)$
5:     $W_M \leftarrow mergeWorldNodes(W_A, W_B)$
6:     $G'_M \leftarrow (G_M + W_M)$
7:     $L_M \leftarrow combineLists(L_A, L_B)$
8:     $PR \leftarrow PageRank(G'_M)$
9:     $L'_M \leftarrow updateScoresList(L_M, PR)$
10:    $update(L_A)$
11:    $update(W_A)$
12:    $Discard(G_M, W_M, L_M, G_B, W_B, L_B)$
```

scalability, since the computations are strictly local and independent of the number of peers in the network, and storage requirements are linear with the number of pages in the network. Moreover, experimental results show that the authority scores given by the algorithm converge to the true global PR scores, as the meetings between peers are performed in the network. Nonetheless, the performance of JXP can be further enhanced, as this Section will show. The extensions concern the meeting step, before the PR computation, where the authority scores from both peers are combined and their local graphs are merged, and the peer selection strategy for choosing a peer for the next meeting.

4.1 Light-weight Merging of Local Graphs

At a peer meeting, instead of merging the graphs and world nodes, we could simply add relevant information received from the other peer into the local world node, and perform the PR computation on the extended local graph and still the JXP scores converge to the global PR scores. The meeting step is then simplified and much more light-weight, as shown by an example in Figure 2.

This has a big impact on the performance, as the graph merging requires considerable computational time; moreover, without the full merging steps, PR is computed for smaller local transition matrices (roughly half the size of the matrices in the full merging). One could argue that the light-weight merging has the drawback of slowing down the convergence speed of the algorithm, since a reduced transition matrix implies a larger number of states that are aggregated on the world node, which could lead to a higher approximation error. This is in fact a valid point, but our experiments never showed any real slow-down of the convergence or bigger errors in comparing JXP scores against true PR scores for the high-ranked pages. Another potential caveat about the light-weight merging could be that the number of iterations for a local PR computation might increase, but again, this never became a real issue in all our experiments.

4.2 Combining Authority Scores

With the new light-weight meeting step proposed, PR is performed at the extended local graph, where the only changes are due to insertion of links from the world node to local pages, whereas links from local pages to the world node are invariant during all iterations of the JXP algorithm. Considering the authority mass transfer, it is intuitive that, from iteration to iteration, more and more authority mass is given to local pages as the peer learns about more incoming links; so the score of the world node should always reduce until the point it is equal to the sum of the true PR scores of the external pages (we will address this property on Sec-

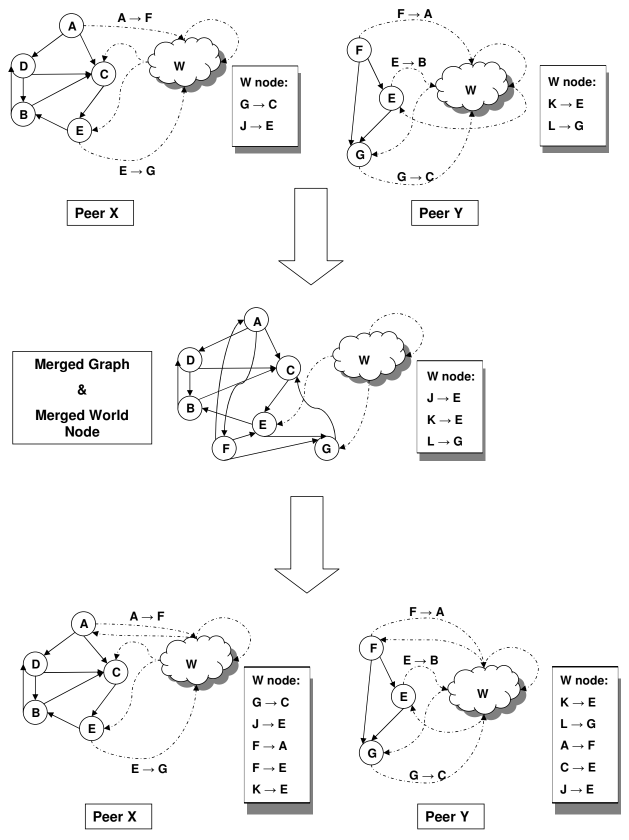


Figure 1: Illustration of the combining and disconnecting procedures.

tion 5, where we proof that this is indeed the case). This is another argument for the convergence of the JXP algorithm.

Based on this consideration, we propose a new way of combining score lists of two peers. Instead of taking the average of the scores of those pages that belong to both lists, we always take the bigger one of the two scores. This is justified by the fact that the world node's score is monotonically non-increasing in the sequence of peer meetings. So we can use a tighter upper bound for the world node's final score to speed up convergence, since a bigger score is an indicator that the peer knows about more incoming links. In addition, when updating the score lists L_A, the scores of pages that do not belong to the local graph G_A should not be re-weighted, as this would result in smaller values, given that the ratio $PR(W)/L_A(W)$ is expected to be always less than one. Thus, the updating procedure is replaced by

$$L'_A(i) = \begin{cases} PR(i) & \text{if } i \in V_A \\ L_A(i) & \text{otherwise} \end{cases} \quad (3)$$

4.3 Peer Selection Strategy

Peers differ in the sets of pages they have indexed, and consequently different peers contribute to a given peer's global view and convergence of scores to different extents. The basic peer selection strategy, where peers are chosen at random, is clearly not the best approach for meeting other peers. Performance could be enhanced if each peer could identify the most promising peers to meet, namely, the ones that would lead to faster convergence of the scores of its locally indexed pages.

A good indicator of the "quality" of a peer, i.e., how much

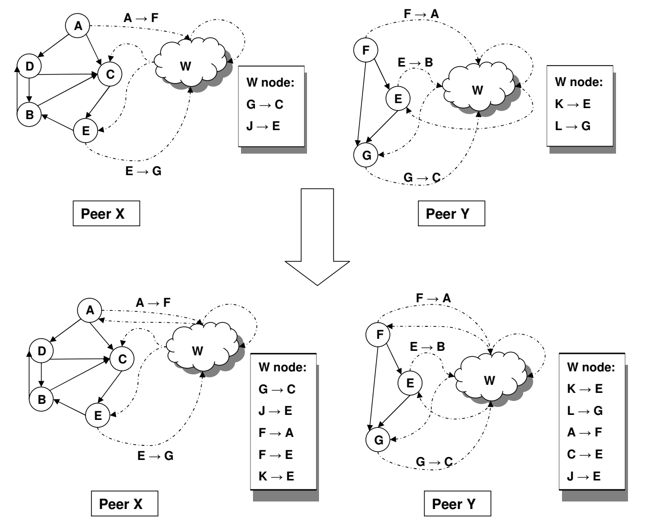


Figure 2: Illustration of the light-weight merging step.

it would contribute to improve another peer's scores, is the amount of outgoing links that are also incoming links for pages in this other peer; the higher the number of links added to the world node, the higher is the amount of authority mass transferred to local pages. The problem now is how to identify these "good" peers, without prohibitively increasing network bandwidth consumption. Our solution is a combination of caching and statistical synopses of the peers' local information.

Peer Synopses

Statistical synopses of peers are a light-weight approximation technique for comparing data of different peers without explicitly transferring their contents. Synopses provide very compact representations for sets, containing some local information that can be used to estimate the correlation between two sets. In comparing sets, we are interested in the measures of "overlap" and "containment". Given two sets, S_A and S_B, the overlap between these two sets is defined as $|S_A \cap S_B|$, i.e., the cardinality of the intersection. The notion of containment was proposed in [9] and is defined as $Containment(S_A, S_B) = |S_A \cap S_B|/|S_B|$. So containment represents the fraction of elements in S_B that are also in S_A.

Fundamentals for statistical synopses of sets have a rich literature, including work on Bloom filters [6, 18], hash sketches [19], and min-wise independent permutations [10]. In this paper we focus on the min-wise independent permutations (MIPs).

The MIPs technique assumes that the set elements can be ordered (which is trivial for integer keys, e.g., hash keys of URLs) and computes N random permutations of the elements. Each permutation uses a linear hash function of the form $h_i(x) := a_i * x + b_i \ mod\ U$ where U is a big prime number and a_i, b_i are fixed random numbers. For each of the N permutations, the MIPs technique determines the minimum hash value, and stores it in an N-dimensional vector, thus capturing the minimum set element under each of these random permutations. By using sufficiently many different permutations, we can approximate the set cardinality and can estimate the containment of two sets.

Pre-meetings Strategy

For the new meeting strategy, we propose that peers perform "pre-meetings", for finding the most promising peers for the next meeting. To this end, we first require all peers to compute two min-wise permutations vectors: one representing its set of local pages, and the other representing the set containing all the successors from all local pages. We call these two MIPs vectors $local(A)$ and $successors(A)$, for a given Peer A.

Assuming that Peer A has chosen Peer B for the next meeting, the pre-meetings strategy works in the following way. During the meeting step, Peer A computes $Containment(successors(B), local(A))$, i.e., that the fraction of local pages in Peer A that has inlinks from local pages in Peer B. If the value is above some pre-defined threshold, Peer A caches Peer B's ID. This way, each peer remembers peers that were previously met and have a relatively high number of inlinks to their local pages. Note that this does not really affect storage requirements, since the threshold limits the number of peers and only the ID of peers are stored.

Still during the meeting step, we also measure the overlap between the local page sets of A and B with the purpose of finding promising peers for a meeting. The idea here is that, given three peers, Peer A, B and C, if Peer C has many links to Peer A, and the overlap between A and B is relatively high, it is very likely that C will have many links pointing to B as well.

Whenever there is a relatively high overlap between two peers, they both exchange their list of cached peers' IDs. The IDs are temporarily stored as potential candidates for a next meeting. For getting the correct correlation with these candidates, pre-meetings are performed with each peer in the temporary list, where instead of exchanging their content, peers return only their MIPs vector representation of their successors sets, $successors(C)$.

The pre-meetings phase does not increase the network load, since only MIPs vectors are sent, and since these vectors are small we can piggyback them on communication messages that are exchanged in the P2P network anyway.

The value $Containment(successors(C), local(A))$ is used to sort peers in the temporary list. Then we select the peer with the highest score on the temporary list for the next, real, meeting (i.e., no longer a pre-meeting), and this step chooses a good candidate with high probability based on our heuristics. After a peer is chosen and the meeting took place, the peer is dropped from this temporary list. It is important that peers have an updated view of the network, as peers can change their contents or eventually leave the network. Therefore, peers have to visit again the already cached peers, with a smaller probability. In addition, the probability of picking a peer at random should never go down to zero, as some peers may not be reachable by merely following the chain of cached peers.

Pseudo code for the optimized version of the JXP algorithm is shown in Algorithm 3. The initialization procedure is the same as the one described previously in Algorithm 1.

5. ANALYSIS OF JXP

In this Section we provide important properties of the JXP scores, as well as a proof for the correctness of the JXP method. We show that JXP scores converge to the correct values, the global PR scores of the individual pages, or equivalently, the stationary visiting probabilities of the underlying global Markov chain. We consider only the optimized JXP version with the light-weight merging from Section 4.1.

Algorithm 3 Optimized JXP Algorithm

```
1: input: local graph $G_A$, world node $W_A$, score list $L_A$
2: repeat
3: $B \leftarrow selectPeer()$
4: $W_A \leftarrow addLinks(G_B, W_B)$
5: $G'_A \leftarrow (G_A + W_A)$
6: $L_A \leftarrow combineLists(L_A, L_B)$
7: $PR \leftarrow PageRank(G'_A)$
8: $update(L_A)$
9: $Discard(G_B, W_B, L_B)$
```

Our analysis builds on the theory of state aggregation in Markov chains [16, 37, 29, 22]. However, applying this theory to our setting is not straightforward at all, and we use it only for particular aspects. State-aggregation techniques assume complete knowledge of the Markov chain and are typically used to speed up the convergence of computations (see, e.g., [25, 14]). In contrast, our P2P setting poses the difficulty that each peer has only limited knowledge of the Web graph and the resulting Markov Model. Moreover, this restricted view differs from peer to peer.

For the proof we assume that there are no changes in the network, so there exists a global web graph with N pages, a global transition matrix $\mathbf{C}_{N\times N}$ and a global stationary distribution vector $\boldsymbol{\pi}$. The element c_{ij} of $\mathbf{C}$ is equal to $1/out(i)$ if there is a link from page i to page j, and 0 otherwise. After adding the random jumps probabilities we have a transition matrix $\mathbf{C}'$

$$\mathbf{C}' = \epsilon\ \mathbf{C} + (1-\epsilon)\frac{1}{N}\mathbf{1}_{N\times N} \tag{4}$$

Every peer has a local graph G, subgraph of the global web graph, that corresponds to the set of pages it has crawled. Pages that are not in G are considered to be on the set $\overline{G}$. The local graph is extended by adding the world node. In our notation a link from page i to page j is represented by $i \rightarrow j$, and W is the set of external pages that are represented in the world node w. For every page r in W we store the information about its outdegree, $out(r)$ and current JXP score $\alpha(r)$, both learned from a previous meeting. The number of local pages is given by n. Associated with each extended local graph we have a local transition matrix $\mathbf{P}$ that has the following format

$$\mathbf{P}_{(n+1)\times(n+1)} = \left(\begin{array}{ccc|c} p_{11} & \dots & p_{1n} & p_{1w} \\ \vdots & \dots & \vdots & \vdots \\ p_{n1} & \dots & p_{nn} & p_{nw} \\ \hline p_{w1} & \dots & p_{wn} & p_{ww} \end{array}\right) \tag{5}$$

where

$$p_{ij} = \begin{cases} \frac{1}{out(i)} & \text{if } \exists\ i \rightarrow j \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

$$p_{iw} = \sum_{\substack{i\rightarrow r \\ r\notin G}} \frac{1}{out(i)} \tag{7}$$

for every i, j, $1 \le i, j \le n$.

The transition probabilities from the world node, p_{wi} and p_{ww}, change during the computation, so they are defining according to the current meeting t

$$p^t_{wi} = \sum_{\substack{r\rightarrow i \\ r\in W^t}} \frac{\alpha(r)^t}{out(r)} \cdot \frac{1}{\alpha^{t-1}_w} \tag{8}$$

$$p^t_{ww} = 1 - \sum_{i=1}^{n} p^t_{wi} \tag{9}$$

For the JXP computation, random jumps are also added, with the particularity that the random jumps to the world node are made proportional to the number of pages it represents. This gives us the following transition matrix

$$\mathbf{P}' = \epsilon\ \mathbf{P} + (1-\epsilon)\frac{1}{N}\mathbf{1}_{(n+1)\times 1}\left(\begin{array}{ccc|c} 1 & \dots & 1 & (N-n) \end{array}\right) \tag{10}$$

which has a stationary distribution vector $\boldsymbol{\alpha}$

$$\boldsymbol{\alpha} = \left(\begin{array}{ccc|c} \alpha_1 & \dots & \alpha_n & \alpha_w \end{array}\right)^T \tag{11}$$

that corresponds to the JXP scores, informally introduced in Section 3 as score lists.

5.1 Initialization Procedure

We start with a local transition matrix, $\mathbf{P}^0$, with all p_{wi} elements equal to zero since the peers start with no knowledge about external pages. The element p_{ww} is consequently set to 1.

$$\mathbf{P}^0_{w*} = \left(\begin{array}{ccc|c} 0 & \dots & 0 & 1 \end{array}\right) \tag{12}$$

The local JXP scores vector is initially set to:

$$\boldsymbol{\alpha}^{init} = \left(\begin{array}{ccc|c} \frac{1}{N} & \dots & \frac{1}{N} & \frac{N-n}{N} \end{array}\right)^T \tag{13}$$

The PR computation is then performed using the transition matrix $\mathbf{P'}^0$ and an updated value for the local authority scores vector $\boldsymbol{\alpha}^0$ $(t = 0)$ is obtained.

5.2 The Meeting Step

As described earlier, the meeting process consists of adding new links, or updating existing links from the world node to the local pages, and performing the PR algorithm using the updated transition matrix.

Consider the follow local transition matrix and its local JXP scores vector at meeting $(t-1)$ $(t \ge 1)$

$$\mathbf{P}^{t-1}_{(n+1)\times(n+1)} = \left(\begin{array}{ccc|c} p_{11} & \dots & p_{1n} & p_{1w} \\ \vdots & \dots & \vdots & \vdots \\ p_{n1} & \dots & p_{nn} & p_{nw} \\ \hline p^{t-1}_{w1} & \dots & p^{t-1}_{wn} & p^{t-1}_{ww} \end{array}\right) \tag{14}$$

$$\boldsymbol{\alpha}^{t-1} = \left(\begin{array}{ccc|c} \alpha^{t-1}_1 & \dots & \alpha^{t-1}_n & \alpha^{t-1}_w \end{array}\right)^T \tag{15}$$

For the sake of simplicity, we split the merging step, by considering only one link addition/update at a time. Assuming that during meeting t a link to page i has been added or updated, we can express p_{wi} at time t as

$$p^t_{wi} = p^{t-1}_{wi} + \delta \tag{16}$$

Since the authority scores of external pages on the meeting step can only increase or remain unchanged we can assure that the value of δ is always non-negative.

As the transition probability from the world node to itself is always adjusted to compensate for changes of the other transition probabilities we can also write

$$p^t_{ww} = p^{t-1}_{ww} - \delta \tag{17}$$

The transition matrix at meeting t can then be written as

$$\mathbf{P}^t = \mathbf{P}^{t-1} + \mathbf{E} \tag{18}$$

where

$$\mathbf{E} = \left(\begin{array}{ccccccc|c} 0 & & & \dots & & & 0 & 0 \\ \vdots & & & \dots & & & \vdots & \vdots \\ 0 & & & \dots & & & 0 & 0 \\ \hline 0 & \dots & 0 & \delta & 0 & \dots & 0 & -\delta \end{array}\right) \tag{19}$$

which leads to an updated JXP scores vector

$$\boldsymbol{\alpha}^t = \left(\begin{array}{ccc|c} \alpha_1^t & \dots & \alpha_n^t & \alpha_w^t \end{array} \right)^T \tag{20}$$

The following two theorems describes important properties about the JXP scores.

Theorem 5.1. *The JXP score of the world node, at every peer in the network, is monotonically non-increasing.*

Proof. The proof is based on the study of the sensitivity of Markov Chains made by Cho and Meyer [15]. From there we can state that by increasing p_{wi} by δ and decreasing p_{ww} by the same amount, the following holds

$$\frac{\alpha_w^{t-1} - \alpha_w^t}{\alpha_w^{t-1}} = \alpha_w^t \ \delta \ m_{iw} \tag{21}$$

where m_{iw} is the mean first passage time from page i to the world node (i.e., the expected number of steps for reaching w when starting in i, in the underlying Markov chain). Rearranging the terms on the equation we have

$$\alpha_w^t - \alpha_w^{t-1} = -\alpha_w^{t-1} \ \alpha_w^t \ \delta \ m_{iw} \tag{22}$$

Since all the values on the right side of the equation are non-negative we can assure that

$$\alpha_w^t - \alpha_w^{t-1} \leq 0 \tag{23}$$

□

Theorem 5.2. *The sum of scores over all pages in a local graph, at every peer in the network, is monotonically non-decreasing.*

Proof. The proof follows from Theorem 5.1 and the fact that the following equality holds

$$\sum_{i \in G} \alpha_i + \alpha_w = 1 \tag{24}$$

□

We now proceed by showing how the JXP scores and the global PR scores are related. The next Theorem shows that the global PR values are an upper bound for the JXP scores.

Theorem 5.3. *Consider the true stationary probabilities (PR scores) of pages $i \in G$ and the world node w, π_i and π_w, and their JXP scores after t meetings α_i^t and α_w^t. The following holds throughout all JXP meetings:*
$0 < \alpha_i^t \leq \pi_i$ for $i \in G$ and $\pi_w \leq \alpha_w^t < 1$.

Proof. We know that for every page $i \in G$:

$$\pi_i = \frac{1-\epsilon}{N} + \epsilon \sum_{\substack{j \to i \\ j \in G}} \frac{\pi_j}{out(j)} + \epsilon \sum_{\substack{j \to i \\ j \in \overline{G}}} \frac{\pi_j}{out(j)} \tag{25}$$

and

$$\alpha_i^t = \frac{1-\epsilon}{N} + \epsilon \sum_{\substack{j \to i \\ j \in G}} \frac{\alpha_j^t}{out(j)} + \epsilon \sum_{\substack{j \to i \\ j \in W^t}} \frac{\alpha_j^t}{out(j)} \frac{\alpha_w^t}{\alpha_w^{t-1}} \tag{26}$$

We prove the claim about the α_i^t values by induction on t; the proof for the claim on the world node follows directly from the fact that the score vector is normalized. The claims that $\alpha_i > 0$ and $\alpha_w^t < 1$ are trivial to show.

For $t = 0$ we consider the situation that a given peer with graph G knows only its local graph and has no information about the world node other than the total number of nodes, N (as explained in Section 5.1). Thus the peer assumes that the only transfer of score mass from w to any node in G is by random jumps, which is the minimum transfer that is possible. Since G includes outgoing links to w, a local PR computation based on this setting cannot overestimate and will typically underestimate the scores of nodes in G.

Now assume that the claim holds for all meetings up to and including t, and consider the $t + 1^{st}$ meeting.

First we observe that because of $\alpha_w^t \leq \alpha_w^{t-1}$ (by Theorem 5.1), $W^t \subseteq \overline{G}$, and the induction assumption $\alpha_j^t \leq \pi_j$, the following upper bound holds for the third summand (abbreviated as β_i):

$$\epsilon \sum_{\substack{j \to i \\ j \in W^t}} \frac{\alpha_j^t}{out(j)} \frac{\alpha_w^t}{\alpha_w^{t-1}} \leq \epsilon \sum_{\substack{j \to i \\ j \in \overline{G}}} \frac{\pi_j}{out(j)} := \beta_i \tag{27}$$

Now consider the following upper bound for α_i^{t+1}:

$$\alpha_i^{t+1} \leq \frac{1-\epsilon}{N} + \epsilon \sum_{\substack{j \to i \\ j \in G}} \frac{\alpha_j^{t+1}}{out(j)} + \beta_i \tag{28}$$

In the $t + 1^{st}$ meeting node i could increase its α_i value in three ways: a) by learning about an additional node $x \in W^{t+1}$ with $x \notin W^t$ that points to i, b) by learning that a previously known node $x \in W^t$ that points to i has a higher value $\alpha^{t+1}(x)$ than the last time that a peer with x in its local graph was met (i.e., at some previous iteration $t' < t+1$), or c) the value α_j^{t+1} of some incoming neighbor j from the peer's own local graph G ($j \in G$) has a higher value than in previous iterations. No other cases are possible.

The last case is impossible unless one of the cases a) or b) occurs, simply because all outdegrees are fixed and, without any external changes, the local PR computation on G will reproduce the scores computed in earlier iterations. But by the induction assumption we have $\alpha_i^t \leq \pi_i$ for all previous t. In the first and second case we can conservatively assume the upper bound β_i for whatever increased score the nodes in W^{t+1} may transfer to i or any other nodes in G. Thus we have

$$\begin{aligned} \alpha_i^{t+1} &\leq \frac{1-\epsilon}{N} + \epsilon \sum_{\substack{j \to i \\ j \in G}} \frac{\alpha_j^{t+1}}{out(j)} + \beta_i \\ &\leq \frac{1-\epsilon}{N} + \epsilon \sum_{\substack{j \to i \\ j \in G}} \frac{\pi_j}{out(j)} + \beta_i = \pi_i \end{aligned} \tag{29}$$

□

Theorem 5.3 does not explicitly reflect the fact that nodes from two local graphs can overlap. We assumed that in these cases the nodes are treated as local nodes, and we take their α_j values from the peer's local bookkeeping. However, because all peers, by Theorem 5.3, invariantly underestimate the true stationary probability of these nodes, we can safely use the maximum of the α_j values from the two peers in a meeting: the maximum is still guaranteed to be upper-bounded by the true PR score π_j.

Theorem 5.3 is a safety property in that it shows that we never overestimate the correct global PR scores. What remains to be done is to show liveness in the sense that JXP makes effective progress towards the true PR scores. The argument for this part is based on the notion of fairness from concurrent programming theory (see, e.g., [24]): a sequence of events is fair with respect to event e if every infinite sequence has an infinite number of e occurrences. In our setting, this requires that in an infinite number of P2P meetings, every pair of peers meet infinitely often. Truly randomized meetings with uniform distribution have this property, but there are other ways as well. A similar argument has been used in [3] for online page importance.

THEOREM 5.4. *In a fair series of JXP meetings, the JXP scores of all nodes converge to the true global PR scores.*

PROOF. The fairness property ensures that at some point, say after the t^{th} meeting, every peer knows all its incoming neighbors, the complete sets $\{j | j \rightarrow i, j \in \overline{G}\}$ for all $i \in G$. At this point, the only reason why a peer's local JXP score α_i^t for some page i may still underestimate the global PR score π_i is that the JXP scores of the incoming neighbors from outside of G may also be underestimated, i.e., $\alpha_j^t < \pi_j$ for some $j \in W$. We show that this situation cannot hold indefinitely, once all the incoming links from external pages are completely known.

There are two cases to consider. The first case is when the world node's JXP score $\alpha_w^{\hat{t}}$ has converged at some point $\hat{t} \geq t$ so that $\alpha_w^{\hat{t}} = \pi_w$ holds (strictly speaking, the difference between the α and the π value is below some ε that can be made arbitrarily small; we simplify the argument for simpler notation). At this point, we can infer that $\sum_{i \in G} \alpha_i^{\hat{t}} = \sum_{i \in G} \pi_i$. So if some $\alpha_i^{\hat{t}}$ is still strictly below its PR score π_i, some other page $j \in G$ must have an $\alpha_j^{\hat{t}}$ value strictly higher than its PR score π_j. But this is impossible because of Theorem 5.3.

The second case is that $\alpha_w^{\hat{t}} < \pi_w$ holds and stays invariant in all subsequent meetings. But then we have $\alpha_w^{\hat{t}+1} = \alpha_w^{\hat{t}}$ which implies:

$$\begin{aligned}\alpha_i^{\hat{t}+1} &= \frac{1-\epsilon}{N} + \epsilon \sum_{\substack{j \rightarrow i \\ j \in G}} \frac{\alpha_j^{\hat{t}+1}}{out(j)} + \epsilon \sum_{\substack{j \rightarrow i \\ j \in \overline{G}}} \frac{\alpha_j^{\hat{t}+1}}{out(j)} \\ &= \frac{1-\epsilon}{N} + \epsilon \sum_{j | j \rightarrow i} \frac{\alpha_j^{\hat{t}+1}}{out(j)}\end{aligned} \quad (30)$$

This is the very same fixpoint equation that we have for the true PR scores, the π_i values. We know that this fixpoint equation has a unique solution [8, 22, 37]; thus the above equation must have the same solution as the equation for the π_i values, and so the JXP scores eventually equal the PR score. (Again, strictly speaking, the difference drops below some ε that can be chosen arbitrarily small.) □

5.3 Additional Considerations

Our convergence proof applies to the optimized, light-weight merging of peer graphs with the local graph extended only by the single world node, and with truly random peer meetings. Also, we assumed that when two peers meet with overlapping graphs, each peer uses its locally stored approximate PR as the estimate for the α_i values. If instead we use the maximum of the two values for pages known at both peers (as advocated in Section 4.2), the convergence proof still holds by the argument given in Theorem 5.3.

As for light-weight merging vs. forming the full union of the graph fragments of two meeting peers, the proof does not carry over to the full-union method. But we not see any compelling reason for not using the light-weight approach. We will show in Section 6.2 on experiments that the accuracy and convergence speed of the light-weight merging are more or less as good as for the full-union method. Thus, we have a convergence proof for the interesting and really relevant method, the light-weight merging.

Peer meeting strategies other than truly random (with uniform choices) could also potentially invalidate the assumptions of the correctness proof. However, all we need to ensure for the proof to hold is that the meeting strategy is fair (in the sense described in Theorem 5.4). This is easy to achieve even with the biased peer selection strategies presented in Section 4.3, simply by making every k^{th} peer selection step truly random. Fairness holds for any constant k, so we can choose a high value for k and primarily pursue the biased meeting strategy.

Finally, we disregarded the dynamics of the P2P network in the sense that we assumed the global graph to be time-invariant. This is unrealistic for various reasons: 1) new Web pages are created, old pages disappear, and links are created or deleted all the time, 2) therefore, peers want to periodically re-crawl parts of the Web according to their interest profiles and refreshing policies, and 3) peers join and leave the P2P network at high rate (the so-called "churn" phenomenon that is typical for P2P networks). Under these conditions, there is no proof of JXP score convergence, and with the current state of the art in P2P computing, there are hardly any guarantees that can be proven under extremely high churn. But this applies also to other, conceptually simpler, properties of P2P systems in general, such as DHT performance guarantees or full correctness under particularly "nasty" failure scenarios [28]. On the positive side, JXP has been designed to handle high dynamics, and the algorithms themselves can easily cope with changes in the Web graph, repeated crawls, or peer churn. Extending the mathematical analysis to include these additional difficulties is a challenge for future work.

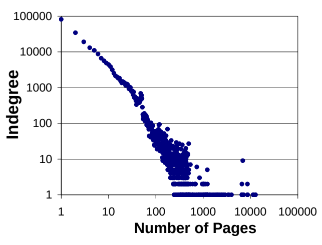
6. EXPERIMENTAL EVALUATION

6.1 Setup

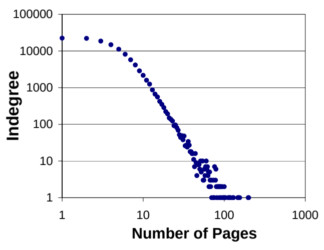
We evaluated the performance of the JXP algorithm on a collection of pages from the Amazon.com website and on a partial crawl of the Web graph. The Amazon data contains information about products (mostly books) offered by Amazon.com. The data was obtained in February 2005, and the graphs were created by considering the products as nodes in the graph. For each product, pointers to similar recommended products are available in the collection. These pointers define the edges in our graphs. Products are also classified into one or more categories. We have thematically grouped together some of the original categories, so in the end we had a total of 10 categories (e.g., "computers", "science", etc).

The Web collection was obtained in January 2005, using the Bingo! focused crawler [36]. We first trained the crawler with a manually selected set of pages and after that, new pages were fetched and automatically classified into one of 10 pre-defined categories such as "sports", "music", etc.

We checked the degree of connectivity to assure that the PR computation was meaningful in these datasets. Figure 3 shows the indegree distribution, on a log-log scale for the two collections. We can see that the two distributions are close to a power-law distribution, which is also the standard assumption for the complete Web graph. We thus expect that our experiments, albeit rather small-scale, are fairly indicative for the behavior at Internet scale.



(a) Amazon Data



(b) Web Crawl

Figure 3: Indegree Distributions.

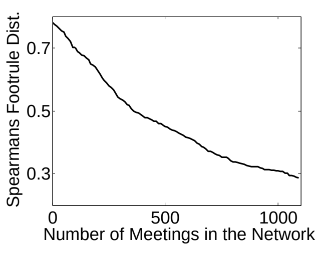
Pages were assigned to peers by simulating a crawler in each peer, starting with a set of random seeds pages from one of the thematic categories and following the links and fetching nodes in a breadth-first approach, up to a certain predefined depth. The category of a peer is defined as the category to which the initial seeds belong. During the crawling process, when the peer encounters a page that does not belong to its category, it randomly decides to follow links from this page or not with equal probabilities. In both of the two setups we have 100 peers, with 10 peers per category. In the Amazon setup there is a total of 55,196 pages and 237,160 links, and in the Web crawl setup we have 103,591 pages and 1,633,276 links. We realize that these are fairly small-scale experiments, but they are nevertheless reasonably indicative. The reason for the limited data volume is that we had to run all 100 peers on a single PC.

6.2 JXP Accuracy And Convergence

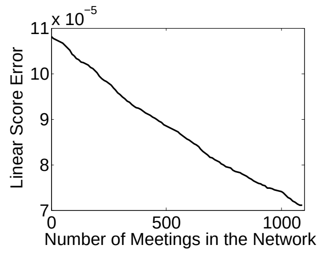
For evaluating the performance we compare the authority scores given by the JXP algorithm against the true PR scores of pages in the complete collection. Since, in the JXP approach, the pages are distributed among the peers and for the true PR computation the complete graph is needed, in order to compare the two approaches we construct a total ranking from the distributed scores by essentially merging the score lists from all peers. (Note that this is done for the experimental evaluation, it would neither be needed nor desired in the real P2P network). We do this periodically after a fixed number of meetings in the network. Since overlaps are allowed and no synchronization is required, it can be the case that a page has different scores at different peers. In this case, the score of the page on the total ranking is considered to be the average over its different scores.

The total top-k ranking given by the JXP algorithm and the top-k ranking given by traditional, centralized PR are compared using Spearman's footrule distance [17], defined as $F(\sigma_1, \sigma_2) = \sum_{i=1}^{k} |\sigma_1(i) - \sigma_2(i)|$ where $\sigma_1(i)$ and $\sigma_2(i)$ are the positions of the page i in the first and second ranking. In case a page is present in one of the top-k rankings and does not appear in the other, its position in the latter is considered to be $k + 1$. Spearman's footrule distance is normalized to obtain values between 0 and 1, with 0 meaning that the rankings are identical, and 1 meaning that the rankings have no pages in common. We also use a *linear score error* measure, which is defined as the average of the absolute difference between the JXP score and the global PR score over the top-k pages in the centralized PR ranking.

First of all, we studied the general behavior of the JXP method, to test whether it serves its purpose as a P2P approximation of global PR. Figures 4 and 5 show Spearman's footrule distance and the linear score error for the Amazon collection and the Web crawl, respectively. Here the scores of the top-1000 highest ranked pages were used, and the charts show the error as a function of the number of peer meetings. We see that the error drops quickly as the peers meet other peers. Already at 1000 meetings the footrule distance drops below 0.3 for the Amazon data and below 0.2 for the Web crawl. At this point, each of the 100 peers, on average, has met and exchanged its graph with 10 other peers. Beyond this point, the JXP scores converge to the global PR values. These observations demonstrate the general viability of the JXP method.

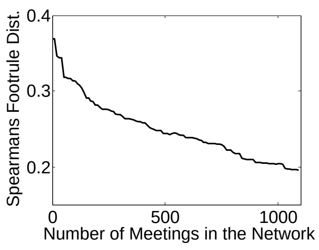


(a)

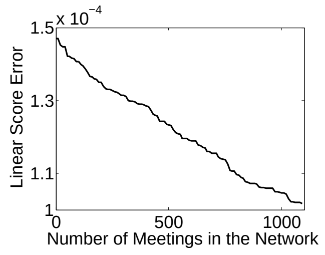


(b)

Figure 4: Spearman's footrule distance 4(a) and linear score error 4(b) for the Amazon data.



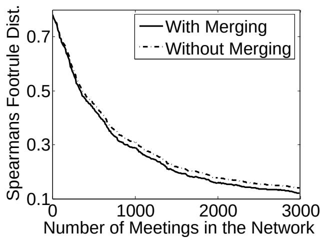
(a)



(b)

Figure 5: Spearman's footrule distance 5(a) and linear score error 5(b) for the Web crawl.

We then evaluated the performance of the proposed lightweight merging procedure against the full merging of the baseline JXP method. The results are also shown in Figures 6 and 7.



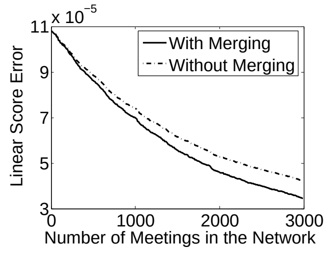


Figure 6: Comparison of merging procedures for the Amazon data.

The charts show that the results are almost unaffected if the graphs are not merged. The small error inserted in the scores did not affect the ranking order of the pages. The performance, however, is highly enhanced, as Table 1 shows. We measured, for each peer, the CPU time (in milliseconds) needed to perform a merging procedure (for one meeting with one other peer). Table 1 presents the average over all meetings a peer has made. Due to space constraints the results are shown only for the three biggest and the three smallest peers (peers were sorted in decreasing order according the their numbers of locally held pages). Similar

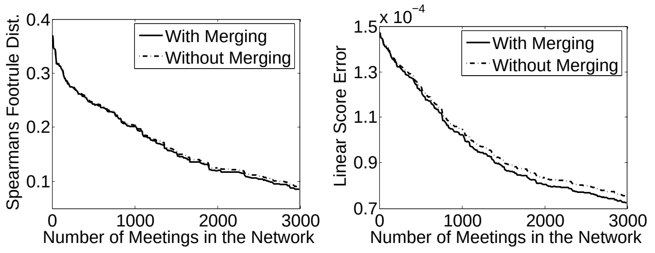


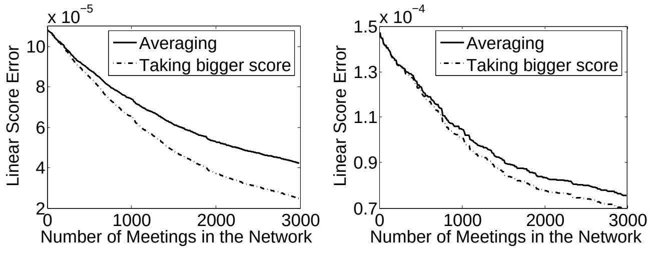
Figure 7: Comparison of merging procedures for the Web crawl.

improvements were obtained for all the other peers as well. As expected, the time needed for the merging procedure drops significantly when we use the light-weight merging.

	Amazon.com		Subset of Web	
	Original Merging	Light-weight Merging	Original Merging	Light-weight Merging
Peer 1	5,505	4,408	31,444	24,943
Peer 2	4,995	4,536	26,024	19,364
Peer 3	3,559	2,233	17,718	13,687
Peer 98	424	166	1,864	229
Peer 99	341	153	1,776	162
Peer 100	269	17	1,403	98

Table 1: CPU time comparison (in milliseconds) between the full merging and the light-weight merging procedures.

Using the light-weight merging procedure, we then compared the performance of the two approaches for combining the score lists. Figure 8 shows the linear score error, where the solid line corresponds to the approach where we first average the scores and then, after the PR computation, reweight the ones corresponding to pages that do not belong to the local graph, and the dashed line is the result for when we always take the bigger score, when combining the lists, and leave the scores of external pages unchanged after the PR computation was performed. Here again, we used the scores of the top-1000 pages.



(a) Amazon data (b) Web crawl

Figure 8: Comparison of the methods for combining the score lists.

The results show that authority scores converge faster to the global PR values when we replace the method for combining the score lists by the one proposed in Section 4.2. They also suggest that the amount of improvement that can be obtained is related to the collection itself. The most interesting and most important improvement, however, is obtained by the peer selection strategy, discussed next.

Figures 9 and 10 present the performance comparison between the two peer selection strategies, with the pre-meetings phase and without the pre-meetings phase, where peers are chosen at random, for the Amazon data and the Web crawl, respectively. For the Web crawl we considered the top-1000 pages, and for the Amazon data we compared the top-10000 pages.

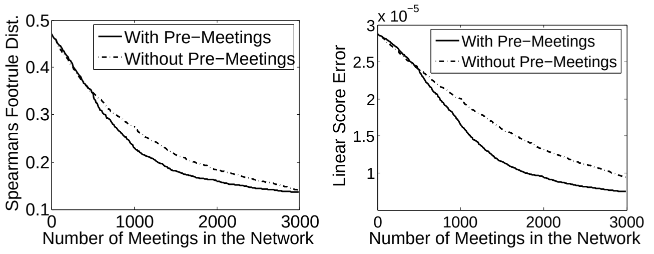


Figure 9: Comparison of peer selection strategies for the Amazon data.

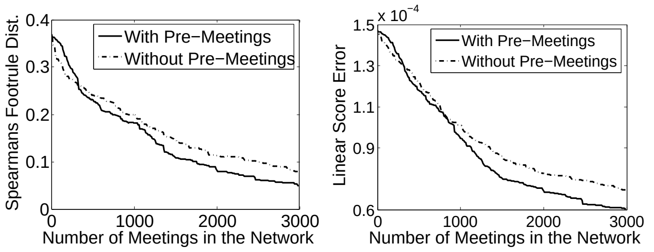
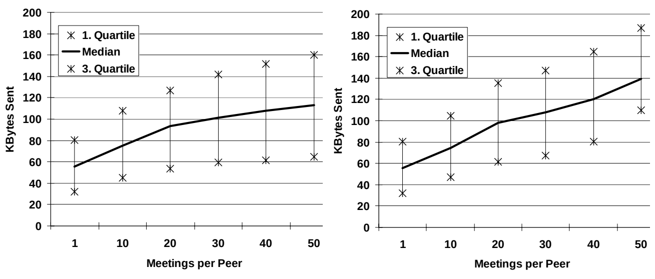


Figure 10: Comparison of peer selection strategies for the Web crawl.

We can see that during the first meetings both approaches perform similarly, but as peers discover, through the pre-meetings, the most promising peers, the number of meetings needed for a good approximation to the global PR scores is reduced. For instance, in the Amazon data, to make the footrule distance drop below 0.2 we needed a total of 1,770 meetings without the pre-meetings phase. With the pre-meetings phase this number was reduced to 1,250. In the Web crawl setup, for a footrule distance of 0.1, the number of meetings was reduced from 2,480 to 1,650. It is clear that the peer selection strategy plays a big role not only on the convergence speed of the JXP algorithm but also on the network load. By finding the most promising peers, many meetings with peers that would contribute only little useful information are avoided.

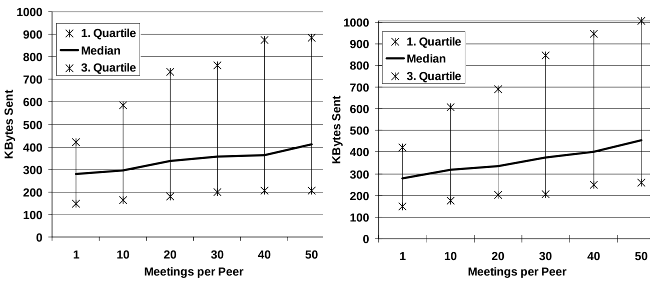
Even though these optimizations significantly reduce the network load, the JXP algorithm still requires a considerable number of meetings. However, the size of the transmitted messages is small, since, for the JXP computation, no page content is required. We measured, for the same setups presented before, the message size of a peer at each meeting. Figures 11 and 12 show the median, the first quartile and the third quartile (in KBytes) for the values at all peers, after each meeting they have performed. We also compare the two peer selection strategies, with and without the pre-meetings phase.

The results show that JXP consumes rather little network bandwidth, as the messages sizes are small. We can also see that the pre-meetings phase causes only a small increase of the number of transmitted bytes, since it requires the exchange of the min-wise independent permutation vectors only. Although the messages transmitted with the pre-meetings phase are slightly bigger, the overall network bandwidth consumption drops significantly, since fewer meetings are performed. For the Amazon data, the total message cost to make the footrule distance drop below 0.2 was around 461MBytes with the pre-meetings phase, compared to the 569MBytes transmitted when meetings were performed at



(a) Without pre-meetings (b) With pre-meetings

Figure 11: Message size (in KBytes) for the Amazon data setup.



(a) Without pre-meetings (b) With pre-meetings

Figure 12: Message size (in KBytes) for the Web crawl setup.

random – a reduction of almost 20%. In the Web crawl, the decrease in the amount of bytes transmitted, for a footrule distance of 0.1, was about 30%, from 4.59 to 3.22 GBytes. We emphasize that these values are the total number of bytes over all meetings performed. Recall that the cost per meeting is small and the time interval between two sucessive meetings can be adapted to the available bandwidth.

6.3 JXP in P2P Search

The JXP algorithm has been integrated into the Minerva system, a prototype platform for P2P Web search under development in our institute [4]. Each Minerva peer is a full-fledged search engine with its own crawler, indexer, and query processor. Peers are autonomous in compiling their own content using a focused Web crawler. A Web query issued by a peer is first executed locally on the peer's own content, and then possibly routed to a small number of remote peers for additional results.

To demonstrate the viability and utility of JXP within the Minerva testbed, we performed a simple and preliminary experiment. Here we have used again our Web collection, but in a different setup. We have created 40 peers out of the 10 categories sets by splitting each set into 4 fragments. Each of the 40 peers hosts 3 out of 4 fragments from the same topic, thus forming high overlap among same-topic peers. In total there were 250,760 documents and 3,123,993 links.

Then we ran 15 queries that are typical for popular Web search requests [7], using the query routing mechanism of Minerva. The merged results were ranked in two ways: 1) by a standard IR model based on term frequency (tf) and inverse document frequency (idf), and 2) by a weighted sum of the tf*idf score and the JXP score (with weight 0.6 of the first component and weight 0.4 of the second component). The queries were taken from [7] and have been intensively used in prior literature on link analysis. We manually (and admittedly somewhat subjectively) assessed the relevance of the top-10 results under the two different rankings. Given the small size of the collection, we considered pages with links to relevant pages not reached by the crawler also as relevant pages. The results for precision at top-10 are given in Table 2. The best results are shown in boldface. On average, the standard tf*idf ranking achieved a precision of 40%, whereas the combined tf*idf/JXP ranking was able to increase precision to 57% percent.

Query	tf*idf	(0.6 tf*idf + 0.4 JXP)
affirmative action	40%	40%
amusement parks	60%	60%
armstrong	20%	**80%**
basketball	20%	**60%**
blues	20%	20%
censorship	**30%**	20%
cheese	40%	**60%**
iraq war	**50%**	30%
jordan	40%	40%
moon landing	**90%**	70%
movies	30%	**100%**
roswell	30%	**70%**
search engines	20%	**60%**
shakespeare	60%	**80%**
table tennis	50%	**70%**
Average	40%	**57%**

Table 2: Precision at top-10 for the Web Collection

7. CONCLUSIONS

We presented the JXP algorithm for dynamically computing authority scores of pages distributed in a P2P network. It runs at every peer, and works by combining locally computed PR scores with meetings among the peers in the network. Through experiments as well as theoretical arguments we showed that the JXP scores converge to the true PR scores that one would obtain by a centralized computation. We also presented a discussion, complemented by experiments results, of optimizations for the algorithm regarding the graph merging procedure and the strategy for selecting a peer for the next meeting. The network bandwidth consumption was also addressed in this work, where we showed that the size of the messages exchanged by the peers is small. In addition, we showed the viability and utility of the algorithm in a P2P search engine, where the result ranking given by the Minerva system was improved by integrating the JXP scores into the score function.

For the future, we plan to address the behavior of the JXP algorithm during changes in the network. We want to analyse how the JXP scores react to addition/removal of peers. Inside the search engine framework, we plan to integrate the JXP scores into the query routing mechanism in order to guide the search for relevant peers for a given query. Finally, a challenging open issue is how to make JXP, P2P search and ranking in general robust in the presence of egoistic, cheating, and malicious peers.

8. REFERENCES

[1] K. Aberer. P-grid: A self-organizing access structure for p2p information systems. In *CoopIS*, 2001.

[2] K. Aberer and J. Wu. A framework for decentralized ranking in web information retrieval. In *APWeb*, pages 213–226, 2003.

[3] S. Abiteboul, M. Preda, and G. Cobena. Adaptive on-line page importance computation. In *WWW Conference*, pages 280–290. ACM Press, 2003.

[4] M. Bender, S. Michel, P. Triantafillou, G. Weikum, and C. Zimmer. Minerva: Collaborative p2p search. In *VLDB*, pages 1263–1266, 2005.

[5] P. Berkhin. A survey on pagerank computing. *Internet Mathematics*, 2(1):73–120, 2005.

[6] B. H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Commun. ACM*, 13, 1970.

[7] A. Borodin, G. O. Roberts, J. S. Rosenthal, and P. Tsaparas. Link analysis ranking: algorithms, theory, and experiments. *ACM TOIT*, 5, 2005.

[8] S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. In *WWW7*, pages 107–117, 1998.

[9] A. Broder. On the resemblance and containment of documents. In *SEQUENCES*, page 21, Washington, DC, USA, 1997. IEEE Computer Society.

[10] A. Z. Broder, M. Charikar, A. M. Frieze, and M. Mitzenmacher. Min-wise independent permutations. *Journal of Computer and System Sciences*, 60(3):630–659, 2000.

[11] A. Z. Broder, R. Lempel, F. Maghoul, and J. Pedersen. Efficient pagerank approximation via graph aggregation.

[12] S. Chakrabarti. *Mining the Web: Discovering Knowledge from Hypertext Data*. Morgan-Kauffman, 2002.

[13] Y.-Y. Chen, Q. Gan, and T. Suel. Local methods for estimating pagerank values. In *CIKM*, pages 381–389. ACM Press, 2004.

[14] S. Chien, C. Dwork, R. Kumar, D. R. Simon, and D. Sivakumar. Link evolution: Analysis and algorithm. *Internet Mathematics*, 1(3):277–304, 2004.

[15] G. Cho and C. Meyer. Markov chain sensitivity measured by mean first passage times. Technical report, NCSU Technical Report #112242-0199, 1999.

[16] P. Courtois. *Decomposability: Queueing and Computer System Applications*. Academic Press, N.Y., USA, 1977.

[17] R. Fagin, R. Kumar, and D. Sivakumar. Comparing top k lists. In *SIAM Discrete Algorithms*, 2003.

[18] L. Fan, P. Cao, J. Almeida, and A. Z. Broder. Summary cache: a scalable wide-area web cache sharing protocol. *IEEE/ACM TON*, 8, 2000.

[19] P. Flajolet and G. N. Martin. Probabilistic counting algorithms for data base applications. *J. Comput. Syst. Sci.*, 31(2):182–209, 1985.

[20] T. H. Haveliwala. Efficient computation of PageRank. Technical report, Stanford University, 1999.

[21] S. Kamvar, T. Haveliwala, C. Manning, and G. Golub. Exploiting the block structure of the web for computing pagerank. Technical report, Stanford University, 2003.

[22] J. G. Kemeny and J. L. Snell. *Finite Markov Chains*. Van Nostrand, Toronto - New York, 1963.

[23] J. M. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 46(5):604–632, 1999.

[24] L. Lamport. *Specifying Systems: The TLA+ Language and Tools for Hardware and Software Engineers*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2002.

[25] A. Langville and C. Meyer. Updating the stationary vector of an irreducible markov chain with an eye on google's pagerank. In *SIMAX*, 2005.

[26] A. N. Langville and C. D. Meyer. Deeper inside pagerank. *Internet Mathematics*, 1(3):335–400, 2004.

[27] C. Lee, G. Golub, and S. Zenios. A fast two-stage algorithm for computing pagerank. Technical report, Stanford University, 2003.

[28] D. Liben-Nowell, H. Balakrishnan, and D. Karger. Analysis of the evolution of peer-to-peer systems. In *PODC*, pages 233–242, 2002.

[29] C. Meyer. *Matrix Analysis and Applied Linear Algebra*. SIAM, Philadelphia, PA, USA, 2000.

[30] J. X. Parreira and G. Weikum. JXP: Global authority scores in a p2p network. In *WebDB*, 2005.

[31] C. Peery, F. M. Cuenca-Acuna, R. P. Martin, and T. D. Nguyen. Wayfinder: Navigating and Sharing Information in a Decentralized World. In *VLDB*, 2004.

[32] S. Ratnasamy, P. Francis, M. Handley, R. M. Karp, and S. Shenker. A scalable content-addressable network. In *SIGCOMM*, pages 161–172, 2001.

[33] A. I. T. Rowstron and P. Druschel. Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In *IFIP/ACM Middleware*, pages 329–350, 2001.

[34] K. Sankaralingam, M. Yalamanchi, S. Sethumadhavan, and J. C. Browne. Pagerank computation and keyword search on distributed systems and p2p networks. *J. Grid Comput.*, 1(3):291–307, 2003.

[35] S. Shi, J. Yu, G. Yang, and D. Wang. Distributed page ranking in structured p2p networks. In *ICPP*, 2003.

[36] S. Sizov, M. Theobald, S. Siersdorfer, G. Weikum, J. Graupmann, M. Biwer, and P. Zimmer. The bingo! system for information portal generation and expert web search. In *CIDR*, 2003.

[37] W. Stewart. *Introduction to the Numerical Solution of Markov Chains*. Princeton University Press, Princeton, NJ, 1994.

[38] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *SIGCOMM*, pages 149–160, NY, USA, 2001.

[39] T. Suel, C. Mathur, J. wen Wu, J. Zhang, A. Delis, M. Kharrazi, X. Long, and K. Shanmugasundaram. Odissea: A peer-to-peer architecture for scalable web search and information retrieval. In *WWW*, 2003.

[40] Y. Wang and D. J. DeWitt. Computing pagerank in a distributed internet search system. In *VLDB*, 2004.

[41] J. Wu and K. Aberer. Using a Layered Markov Model for Distributed Web Ranking Computation. In *ICDCS*, 2005.