The VLDB Journal (2008) 17:291–313
DOI 10.1007/s00778-007-0057-y

SPECIAL ISSUE PAPER

The Juxtaposed approximate PageRank method for robust PageRank approximation in a peer-to-peer web search network

Josiane Xavier Parreira · Carlos Castillo · Debora Donato · Sebastian Michel · Gerhard Weikum

Received: 16 February 2007 / Accepted: 22 April 2007 / Published online: 26 June 2007

Abstract We present Juxtaposed approximate PageRank (JXP), a distributed algorithm for computing PageRank-style authority scores of Web pages on a peer-to-peer (P2P) network. Unlike previous algorithms, JXP allows peers to have overlapping content and requires no a priori knowledge of other peers' content. Our algorithm combines locally computed authority scores with information obtained from other peers by means of random meetings among the peers in the network. This computation is based on a Markov-chain state-lumping technique, and iteratively approximates global authority scores. The algorithm scales with the number of peers in the network and we show that the JXP scores converge to the true PageRank scores that one would obtain with a centralized algorithm. Finally, we show how to deal with misbehaving peers by extending JXP with a reputation model.

Keywords Link analysis · Web graph · Peer-to-peer systems · Social reputation · Markov chain aggregation

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

J. X. Parreira (✉) · S. Michel · G. Weikum
Max-Planck Institute for Informatics, Saarbrücken, Germany
e-mail: jparreir@mpi-inf.mpg.de

S. Michel
e-mail: smichel@mpi-inf.mpg.de

G. Weikum
e-mail: weikum@mpi-inf.mpg.de

C. Castillo · D. Donato
Yahoo! Research, Barcelona, Spain
e-mail: chato@yahoo-inc.com

D. Donato
e-mail: debora@yahoo-inc.com

1 Introduction

1.1 Motivation and problem

Peer-to-peer 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) [60]. Another interesting 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 (e.g., [8,10,24,35,53,63]) and could offer various advantages: (i) lighter load and smaller data volume per peer, and thus more computational resources per query and data unit, enabling more powerful linguistic or statistical learning methods; (ii) 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; (iii) a decentralized approach could provide better immunity to search result distortion by the bias of big providers, commercial interests, or even censorship.

Social communities is another concept that has lately been explored to improve the search experience (e.g., del.icio.us, flickr.com). With billions of people from different parts of the world contributing with their input, the task of identifying the "hot spots" of a community becomes crucial. The community users interact in a way that results in community graphs that allow authority analyses similar to popular PageRank-style analyses on Web graphs [19]. Such community graphs naturally arise in various applications, by different means of user interaction, with respect to a wide variety of entities, and with varying notions of authority (e.g., product ratings, opinions

on other people' blogs or photos, bibliographic references, etc.).

PageRank-style authority scoring, based on the Eigenspace analysis of a suitably defined graph of Web links, endorsements, or interactions, is an established tool for ranking information units (Web pages, sites, peers, social groups, etc.) by their relative importance [13,19,44]. As Google has impressively demonstrated, such authority information can be exploited for improved ranking of search results.

Recently, various techniques have been proposed for speeding up these analyses by distributing the link graph a-mong multiple sites [2,36,64]. 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 [36]. 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.

In this paper we address the problem of computing PR in a general P2P setting with potentially overlapping graph fragments distributed across peers of a large network. 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 analysis. 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.

1.2 Contribution and outline

JXP (Juxtaposed approximate PageRank) is an algorithm for coping with the above situation: dynamically computing, in a decentralized P2P manner, global authority scores when the Web graph is spread across many autonomous peers with arbitrarily overlapping graph fragments and the peers are a priori unaware of other peers' fragments. 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 a mathematical framework for the analysis of JXP, where important properties are derived and it is proven that the JXP scores converge to the true global PR scores. Applications of the algorithm are also given, where we have integrated the JXP scores into a P2P search engine in order to improve the ranking of the results.

Since high authority scores can bring benefits for peers, it is expected that malicious peers would try to distort the correctness of the algorithm, by providing different (usually higher) scores for some of their local pages. P2P networks are generally vulnerable to malicious agents that can cheat in order to get more benefits. [48] points out that P2P architectures for information sharing, search, and ranking must integrate a complete *reputation system*. Reputation systems operate by collecting information on the behavior of the peers, scoring each peer based on good versus bad behavior, and allowing the system to take countermeasures against suspicious peers.

In this work we also present a trust model that integrates decentralized authority scoring with an equally decentralized reputation system. Our approach is based on anomaly detection techniques that allow us to detect a suspicious peer based on the deviation of its behavior from some common features that constitute the usual peer profile. Our method combines an analysis of the authority score distribution and a comparison of score rankings for a small set of pages. The JXP algorithm is then enhanced to avoid the impact of malicious peers. We call this enhanced version *TrustJXP*.

Preliminary results of this research have been presented in [51,52]. The current paper extends these prior conference publications in the following ways: (1) We provide additional detail in the mathematical analysis of the JXP algorithm, including a discussion of its robustness to misestimated input parameters, most notably, the estimate of the total number of Web pages, as well as an experimental indication of the scalability of the algorithm. (2) We provide methods for making JXP robust to dishonest peers that aim to manipulate the data that is exchanged among peers at peer meetings, and we show the effectiveness of the techniques for detecting and compensating various forms of misbehavior in experiments.

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 Sect. 3. The extensions and run-time improvement of JXP are discussed at Sect. 4, and the mathematical analysis is given at Sect. 5. The trust

model and the TrustJXP algorithm are presented at Sect. 6. Experimental results are described in Sect. 7, and Sect. 8 concludes the paper with ideas for future work.

2 Related work

2.1 Link analysis

Link-based authority ranking has received great attention in the literature. It has started with the seminal works of Brin and Page [14] and Kleinberg [41], and after these, many other models and techniques have followed. Good surveys of the many improvements and variations are given in [11,13,19,45].

2.1.1 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:

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

where N is the total number of pages in the link graph, $\mathrm{PR}(p)$ 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.

2.1.2 Distributed PageRank

With the advent of P2P networks [1,54,55,62] attention to distributed link analysis techniques has been growing.

In [64], 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 [66] 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. [36] that postulates a block structure of the link matrix and exploits this structure for faster convergence of the global PR computation. A fundamental approach to distributed spectral decomposition of graphs is given by Kempe and McSherry [40], where distributed link analysis would be a special case of the presented mathematical and algorithmic framework. Related methods that can compute authority and centrality measures on a variety of directed and undirected graph structures are given by Canright et al. [18]. All these methods require and build on particular distribution of pages among the sites where the graph fragments have to be *disjoint*, which makes them suitable for certain classes of distributed systems and also for accelerating link analysis on a cluster of computers, but less attractive for a P2P environment. In a P2P network, disjoint partitioning would be 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. [20] 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 [21,43] 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 et al. [17], 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.

Sankaralingam et al. [56] 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. [57] 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, the JXP algorithm performs the actual computations locally at each peer, and thus needs a much smaller number of messages.

2.2 Trust models

A general framework for different types of trust and distrust propagation in a graph of Web pages, sites, or other entities is introduced in [30]. TrustRank [33] is a PageRank-like authority measure based on manually labeling a seed of highly trusted hosts, and then propagating that trust to other hosts. This algorithm allows estimating the amount of trusted score that each Web page receives and indirectly evaluating also the amount of score received by spammers. In [31,65], the original TrustRank idea has been further extended.

Detecting and combating Web link spam is a special, but highly important case of reasoning about trust and distrust. [32] gives a taxonomy of the most important spamming techniques. A number of algorithms have been presented in order to fight spam. Most of them (see, e.g., [6,7,26,65,31]) analyze the statistical properties of the link structure induced by the Web and classify as spam those pages that exhibit statistically significant local deviations in these properties.

The problem of untrustworthy or manipulated content is felt even more in a P2P environment [60]. The complete lack of accountability of the resources that peers share on the network offers an almost ideal environment for malicious peers and forces the introduction of reputation systems that help to assess the quality and trustworthiness of peers. In [48], the authors present a complete overview of the issues related to the design of a decentralized reputation system. EigenTrust [37] is one of the first methods introduced to assign a global trust value to each peers, computed as the stationary distribution of the Markov chain defined by the normalized local trust matrix C where c_{ij} is the local trust value that a peer j assign to a peer j. Extensions towards distributed and non-manipulable EigenTrust computations are presented in [4]. The anomaly detection procedure described in [59] analyzes peer activity on the network in order to identify peers whose behavior deviates from the typical peer-traffic profile (e.g., duration of connections, uploaded bytes, etc.). In [50] the authors present SeAl, an infrastructure designed for addressing the problem of selfish peer behavior. It works by combining a monitoring/accounting subsystem, an auditing/verification subsystem, and incentive mechanisms. Another framework for reputation-based trust management is presented in [67], where peers give feedback about other peers' good or bad behavior and various forms of network-wide trust measures can be derived in a P2P-style distributed computation.

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 a-mong 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 $\leftarrow$ 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 we show later in this paper that a wrong estimate of the number global graph size only causes a rescaling on the JXP scores, while the ranking order of the pages is preserved.

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 exchanges 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 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:

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

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.

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: $Discard(G_M, W_M, L_M, G_B, W_B, L_B)$

4 Extensions and optimizations

The JXP algorithm, as presented before, already has nice scalability, since the computations are strictly local and independent of the number of peers in the network, and storage requirements per peer are linear in the number of locally hosted pages. Moreover, we show, both mathematically and experimentally, 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.

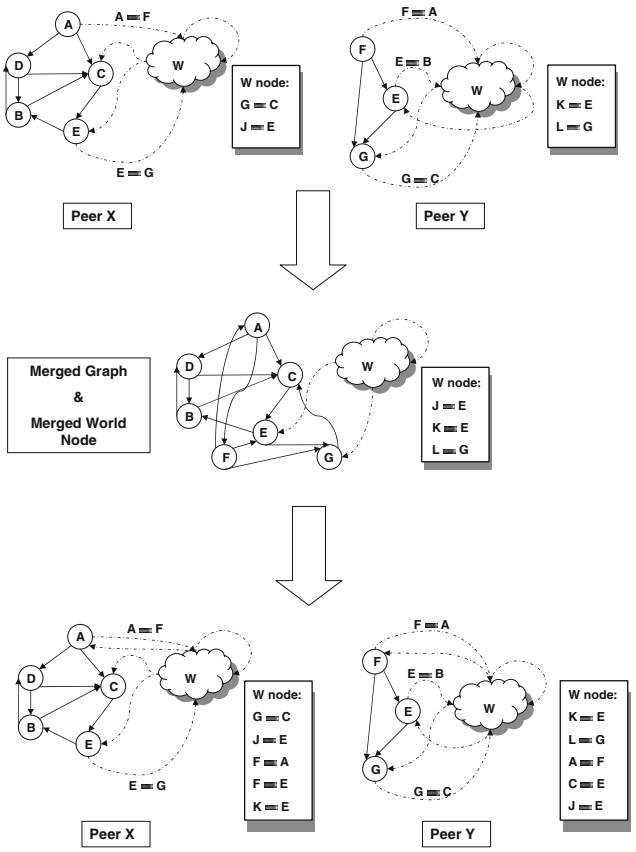

Fig. 1 Illustration of the combining and disconnecting procedures

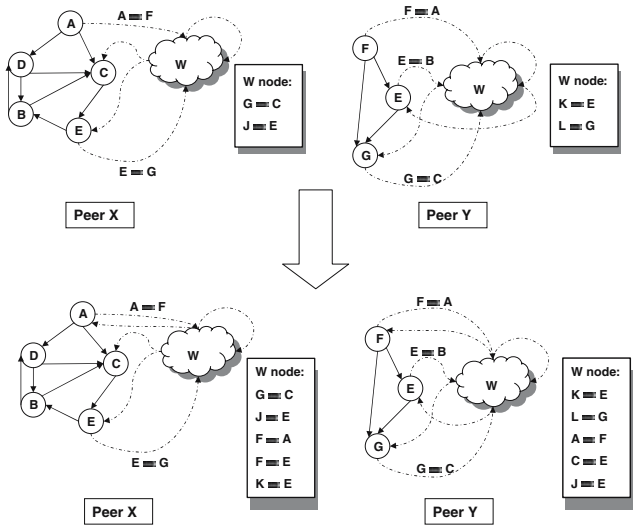

Fig. 2 Illustration of the light-weight merging step

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 Fig. 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 Sect. 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 $\mathrm{PR}(W)/L_A(W)$ is expected to be always less than one. Thus, the updating procedure is replaced by

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

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 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 lightweight 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 [15] 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 [12,28], hash sketches [29], and min-wise independent permutations [16]. In this paper we focus on 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 has been chosen and the meeting has taken 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.

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: $Discard(G_B, W_B, L_B)$

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 Sect. 4.1.

Our analysis builds on the theory of state aggregation in Markov chains [23,38,49,61]. 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., [21,43]). 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

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

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, $1 \leq i, j \leq n$.

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

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

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

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

$$\mathbf{P}' = \epsilon\ \mathbf{P} + (1-\epsilon)\frac{1}{N}\mathbf{1}_{(n+1)\times 1}\left(1\ \ldots\ 1 \middle| (N-n)\right) \tag{10}$$

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

$$\boldsymbol{\alpha} = \left(\alpha_1\ \ldots\ \alpha_n \middle| \alpha_w\right)^T \tag{11}$$

that corresponds to the JXP scores, informally introduced in Sect. 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.

$$\mathbf{P}^0_{w*} = \left(0\ \ldots\ 0 \middle| 1\right) \tag{12}$$

The local JXP scores vector is initially set to:

$$\boldsymbol{\alpha}^{init} = \left(\tfrac{1}{N} \ \ldots \ \tfrac{1}{N} \middle| \tfrac{N-n}{N} \right)^T \tag{13}$$

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 $(t-1)$ $(t \geq 1)$

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

$$\boldsymbol{\alpha}^{t-1} = \left(\alpha^{t-1}_1 \ \ldots \ \alpha^{t-1}_n \middle| \alpha^{t-1}_w \right)^T \tag{15}$$

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

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

which leads to an updated JXP scores vector

$$\boldsymbol{\alpha}^t = \left(\alpha^t_1 \ \ldots \ \alpha^t_n \middle| \alpha^t_w \right)^T \tag{20}$$

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

Theorem 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 [22]. From there we can state that by increasing p_{wi} by δ and decreasing p_{ww} by the same amount, the following holds

$$\frac{\alpha^{t-1}_w - \alpha^t_w}{\alpha^{t-1}_w} = \alpha^t_w \, \delta \, m_{iw} \tag{21}$$

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

$$\alpha^t_w - \alpha^{t-1}_w = -\alpha^{t-1}_w \, \alpha^t_w \, \delta \, m_{iw} \tag{22}$$

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

$$\alpha^t_w - \alpha^{t-1}_w \leq 0 \tag{23}$$

Theorem 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 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 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^t_i and α^t_w. The following holds throughout all JXP meetings: $0 < \alpha^t_i \leq \pi_i$ for $i \in G$ and $\pi_w \leq \alpha^t_w < 1$.*

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 Sect. 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 $t + 1^{\text{st}}$ meeting.

First we observe that because of $\alpha_w^t \leq \alpha_w^{t-1}$ (by Theorem 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 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 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 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., [42]): 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 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 tth meeting, every peer knows all its incoming neighbors, the complete sets $\{j | j \to i, j \in \overline{G}\}$ 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 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:

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

$$= \frac{1-\epsilon}{N} + \epsilon \sum_{j|j \to i} \frac{\alpha_j^{\hat{t}+1}}{out(j)} \quad (30)$$

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 [14,38,61]; 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 Sect. 4.2), the convergence proof still holds by the argument given in Theorem 3.

As for light-weight merging versus 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 do not see any compelling reason for not using the light-weight approach. We will show in Sect. 7.2 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 4). This is easy to achieve even with the biased peer selection strategies presented in Sect. 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 [47]. 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.

5.4 Misestimation of the global number of pages

The JXP algorithm assumes knowledge of the total number of distinct pages in the P2P network in order to compute the random jumps probabilities and correctly converge to the global PageRank values. Although there are efficient techniques for distributed counting with duplicate elimination [9,34,39], the need for knowing this global quantity could be a problem.

Our studies have found that the true value of the number of pages in the network is only needed when we are interested in the correct stationary distribution values for each page. For cases where the correct values is not a must, as long as the *ranking* is correct, which is often the case, any choice for the random jump probability is sufficient, as long as the value for the global number of pages is the same across all peers and greater than the largest local collection.

To formalize this result about different values for computing the random jumps probabilities we redefine the transition matrix from Eq. (10) as follows

$$\mathbf{P}'(\mathbf{X}) = \epsilon\, \mathbf{P} + (1-\epsilon)\frac{1}{X}\mathbf{1}_{(n+1)\times 1}\left(1 \ldots 1 \middle| (X-n)\right) \quad (31)$$

where X is the value used to replace the global number of pages N. When N is known, we have $X = N$ and the results are the same as given on the previous subsections.

The convergence of the JXP algorithm for different choices of X is guaranteed by the following theorem.

Theorem 5 *The JXP local transition matrices, at every peer, are always stochastic, for any choice of $X > n$.*

Proof By inspection of the matrix $\mathbf{P}'(\mathbf{X})$ we can see that it satisfies all three conditions for being stochastic [61]

1. $p'_{ij} \geq 0$ for all i, j
2. $\sum_j p'_{ij} = 1$ for all i
3. At least one element in each column differs from zero.

The first and third conditions require that $X > n$.

Theorem 5 guarantees that there exists a stationary distribution vector $\boldsymbol{\alpha}(\boldsymbol{X})$

$$\boldsymbol{\alpha}(\boldsymbol{X}) = \left(\alpha_1(X) \ldots \alpha_n(X) \middle| \alpha_w(X)\right)^T \tag{32}$$

associated with each local matrix, which corresponds to the JXP scores.

Although this result does not mathematically relate the $\alpha_i(X)$ values with the π_i values, our experiments (see Sect. 7) indicate that $\alpha_i(X)$ values, with $X \neq N$ are related to $\alpha_i(N)$ by a scaling factor, which results in the ranking orders to remain unchanged.

6 Identifying Malicious behavior

The decentralized nature of the JXP computation relies on the information given by each peer in order to compute the global authority scores. However, it is known that in such decentralized environment peers might cheat in an attempt to distort the scores by providing manipulated data in the peer meetings. In this section we propose an enhanced version of the JXP, that contains a variety of statistical techniques for detecting suspicious behavior. The enhanced version, coined TrustJXP, is again completely decentralized, and we demonstrate its viability and robustness in experimental results (see Sect. 7). TrustJXP does not require any form of cooperation among peers. Peers may be arbitrarily egoistic or malicious, but we assume that the fraction of well behaving peers is significantly larger than the fraction of cheating peers. TrustJXP can also operate with anonymous peers.

There are many possible forms of attacks or manipulations in a P2P network.

In this paper we focus on attacks where peers want to distort the authority scores being computed by JXP, by reporting, whenever asked by another peer, wrong scores for a set of pages. We have modeled two types of such score manipulation:

1. A cheating peer can report a higher score for some or all of its local pages, in an attempt to get its pages into high positions in the global ranking that JXP peers may perceive. In this form of manipulation, the peer would boost pages at the "expense" of reducing the total weight of its world node (giving lower score mass to all non-local pages).
2. A cheating peer can manipulate the scores of its local pages by permuting the scores. This way, some pages are boosted while others are downgraded. The score mass of the world node would stay unchanged. Moreover, the statistical distribution of the scores among local pages would appear identical to the original distribution. So this attack is harder to detect.

In the following subsections we describe how we detect and eliminate or compensate the effects of such attacks.

6.1 Malicious increase of scores

As we mention earlier, having documents with high authority scores can bring many benefits for the peer: with its pages appearing at the first positions in the ranking for answering queries posted on the network, the probability that a user clicks on one of them is higher, which may translate, for instance, in revenue for that peer.

To combat this kind of manipulation we use the scores distributions of the pages in a peer's local graph. After a few iterations, the local distributions should resemble the global distribution. The justification for this hypothesis stems from the way the local graph fragments are built. In our P2P model, each peer gathers its data by performing Web crawls, starting from particular seeds and possibly using a thematically focused crawler in order to harvest pages that fit with the interest profile of the corresponding user (or user group). Given that the Web graph is self-similar [5,25], the properties of the small graph fragment that a peer eventually compiles should be statistically indistinguishable from the properties of the full Web graph as seen by a centralized crawler. Dill et al. [25] observed these properties also across different partitions of the Web graph, including the case where pages were separated by their content, which corresponds to using a focused crawler. We use histograms for storing and comparing the different scores distributions.

Histograms Histograms provide a compact representation of the scores distributions. Pages are assigned to histogram buckets according to their JXP scores. Since scores are expected to follow a power-law distribution, we make the boundaries of the buckets also exponential, similar to what is used in [7]. More precisely, the bucket number i will have the boundaries

$$bucket(i) = [a \cdot b^{i-1}, a \cdot b^i)$$

where the values for a and b are 0.005 and 0.3, respectively. We chose these values in order to cover the range of expected values for the scores. The precise values for a

and b will depend on the distribution of PageRank values in the observed sample, which in turn depends on the number of pages in the entire network and the dampening factor.

We create, at each peer, a histogram which is initially filled with the initial JXP scores of local pages. After each meeting, the distribution of the local scores of the other peer is added to the local histogram. We introduce a *novelty factor* to account for the dynamics of the scores across the meetings. Given the local histogram at meeting t, H^t, and the score distribution from the other peer D, the local histogram at meeting $(t+1)$ is updated as follows:

$$H^{(t+1)} = (1-\rho)H^t + \rho D$$

where the parameter ρ represents how much importance we give to the new values. In our framework we set $\rho = 0.6$.

Since we rely on the assumption that the number of honest peers is significantly bigger than the number of dishonest ones, we expect that the histogram always reflects the true distribution of the honest peers. If dishonest peers are reporting higher scores for some of their local pages, the distribution of their local scores would no longer resemble the distribution expected over all peers. Therefore, a comparison against the accumulated local histogram should give an indication of this deviation from normal behavior. How we compare the distributions is explained in the next subsection.

Comparing histograms Given the accumulated local histo-gram of a peer i, H_i, and the histogram containing the scores distribution of another peer j, D_j, we want to compute how much D_j deviates from H_i. Since the distributions are expected to be similar [25], we believe that the distributions of honest peers should be very close to each other, and if D_j differs from H_i by a large margin, it is an indication that the peer is cheating about its local scores. For comparing the two distributions we have chosen the *Hellinger Distance*, which is defined as [46]:

$$\mathrm{HD}_{i,j} = \frac{1}{\sqrt{2}}\left[\sum_k (\sqrt{H_i(k)} - \sqrt{D_j(k)})^2\right]^{\frac{1}{2}} \tag{33}$$

where, k is the total number of buckets and $H_i(k)$ and $D_j(k)$ are the number of elements at bucket k at the two distributions, both normalized by the total number of elements at each distribution. The factor $1/\sqrt{2}$ is introduced to normalize to range of possible values.

As an alternative to the Hellinger Distance, we could also use the χ^2 goodness-of-fit test or information-theoretic measures such as Kullback-Leibler divergence. We implemented all variants, and found that the Hellinger Distance gave the most robust results, but the other methods worked fine, too. Since it is a metric, Hellinger Distance has nice properties, besides the fact that values can be normalized, which makes it easier to be combined with other measures.

6.2 Malicious permutation of scores

Our histograms comparison is inherently unable to detect a cheating peer that reports a permutation of the current scores of its local pages, since both distributions would be statistically indistinguishable. For detecting this type of attack we use a different technique. In our experimental studies of the JXP algorithm, we have observed that, after a few meetings, although the local JXP scores do not correspond yet to the global authority scores, peers already start having a good notion of the relative ranking of their local pages. Given this fact, a comparison of the relative ranking of pages in both local graphs should give us hints about cheating attempts.

We compare the two rankings of the two peers in a meeting for those pages fall into the overlap of both local graphs, and we measure what we refer to as the *Tolerant Kendall's Tau Distance*, defined below.

We use a relaxation of Kendall's Tau since we need to tolerate small fluctuations in the scores of pages with almost identical global authority. To this end, we discount page pairs that have different relative orders in the two rankings if their score differences are below a tunable threshold Δ. In this case, we consider the page pair as incomparable and their rank order as arbitrary.

Our Tolerant Kendall's Tau Distance is therefore defined as:

$$\begin{aligned} K'_{i,j} = |(a,b) : a < b \wedge score_i(a) - score_i(b) \geq \Delta \\ \wedge \tau_i(a) < \tau_i(b) \wedge \tau_j(a) > \tau_j(b)| \end{aligned} \tag{34}$$

where $score_i(a)$ and $score_i(b)$ are the scores of pages a and b at peer i, $a < b$ refers to the lexicographical order of page URLs (to avoid double-counting), τ_i and τ_j are the rankings of pages in the overlapping set at peers i and j, and Δ is our tolerance threshold. A good choice of Δ can be derived from the dampening factor of the underlying PageRank model as follows. We consider as our threshold the minimum amount of authority mass one page can have, which is the score mass earned from the random jumps. Therefore, at each peer, Δ is set to

$$\Delta = \frac{(1-\epsilon)}{N} \tag{35}$$

where ϵ is usually set to 0.85 and N is the total number of pages in the network.

This approach assumes that whenever two peers meet, there is a sufficient overlap between their locally known pages to make this comparison statistically meaningful. In an application where such overlaps cannot be guaranteed with high probability, we would have to add artificial overlaps as

"honesty witnesses". One way of designing such an additional set of witness pages would be to randomly draw a set of sample URLs and disseminate them in the network by an epidemic protocol or using the overlay network of the P2P system. This set of witnesses should be changed periodically to counter adaptation strategies of malicious peers.

6.3 Computing trust scores

We now use our trust model to assign trust scores to peers. The method is totally decentralized: each peer is responsible for assigning (its perception of) trust scores to other peers, based on interactions with them. During a meeting, peers exchange the scores of their local pages. These scores are used for computing both histograms divergence and the rank divergence for the overlapping pages. These two measures will determine the level of trust that should be given to the peer. A new trust score is assigned to a peer at every meeting, as scores are changing.

It is important to emphasize that our technique relies on comparing only the scores of the local pages without any further information about peer identity. This characteristic makes the algorithm resilient to simple Sybil attacks where a single "bad peer" is banned from the network but then re-joins it under a new identity. TrustJXP never considers the identities of peers, and thus has certain immunity against this simple form of Sybil attacks.

For combining histograms divergence and rank divergence into one single trust score, we take a conservative choice: we always take the lower level of trust among the two measures. Thus, we define the trust score that a peer i gives to a peer j as

$$\theta_{i,j} = \min(1 - \mathrm{HD}_{i,j}, 1 - K'_{i,j}) \tag{36}$$

This is the trust score that will be used in the TrustJXP algorithm, which is presented in the following section.

6.4 Integrating trust scores and JXP scores

TrustJXP incorporates the trust measure θ into the JXP algorithm for computing more reliable and robust authority scores. Our approach is to use the trust measure at peer meetings when combining the scores lists. For combining the scores lists, in the JXP algorithm, whenever a page is present in both lists, its score will be set to the average of both scores or the maximum of the two scores, depending on the approach chosen. More formally, the score of page i in the updated score list L' is given by

$$L'(i) = \begin{cases} (L_A(i) + L_B(i))/2 & \text{if "average"} \\ \max(L_A(i), L_B(i)) & \text{if "maximum"} \end{cases} \tag{37}$$

where $L_A(i)$ and $L_B(i)$ are the scores of page i at the two peers. If the page is not in one of the lists, its value is set to zero on the respective list.

For the TrustJXP algorithm, the contribution of the scores from the other peer are weighted based on how much that peer is considered to be trustworthy. The score of a page i in the updated scores list is now defined as

$$L'(i) = \begin{cases} (1-\theta/2) * L_A(i) + \theta/2 * L_B(i) & \text{if "average"} \\ \max(L_A(i), \theta * L_B(i)) & \text{if "maximum"} \end{cases} \tag{38}$$

After combining the scores lists, the JXP algorithm proceeds as usual: the relevant information learned from the other peer is added to the world node, and a PR computation is performed, leading to new JXP scores.

7 Experimental evaluation

7.1 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 ten categories (e.g., "computers", "science", etc).

The Web collection was obtained in January 2005, using the Bingo! focused crawler [58]. 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 ten 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.

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

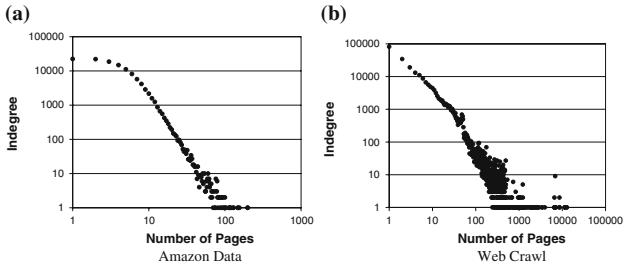

Fig. 3 Indegree distributions

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. For both datasets we have 100 peers, with 10 peers per category. In the Amazon setup there is a total of 52,639 pages and 221,380 links, and in the Web crawl setup we have 134,405 pages and 1,915,401 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.

7.2 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 [27], 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 additionally consider 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-1,000 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 1,500 meetings the footrule distance drops below 0.4 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 15 other peers. Beyond this point, the JXP scores converge to the global PR values. These observations demonstrate the general viability of the JXP method.

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

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

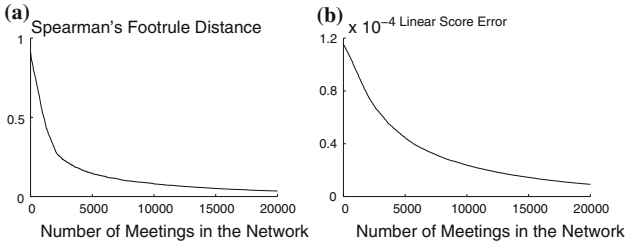

Fig. 4 Spearman's footrule distance (**a**) and linear score error (**b**) for the Amazon data

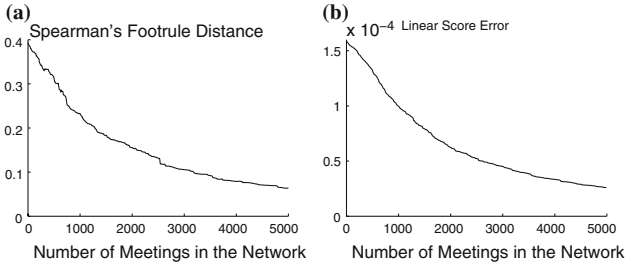

Fig. 5 Spearman's footrule distance (**a**) and linear score error (**b**) for the Web crawl

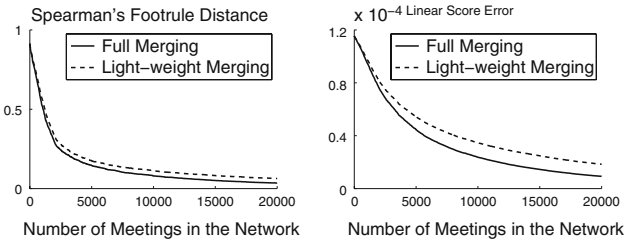

Fig. 6 Comparison of merging procedures for the Amazon data

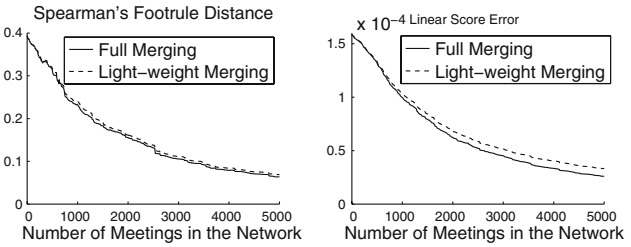

Fig. 7 Comparison of merging procedures for the Web crawl

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

	Amazon.com		Subset of Web	
	Original merging	Light-weight merging	Original merging	Light-weight merging
Peer 1	2,480	853	60,309	31,270
Peer 2	2,337	813	51,380	22,508
Peer 3	2,230	648	49,680	22,336
Peer 98	756	87	6,303	175
Peer 99	725	76	5,734	167
Peer 100	683	56	4,244	103

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 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.

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, re-weight 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-1,000 pages.

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 Sect. 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.

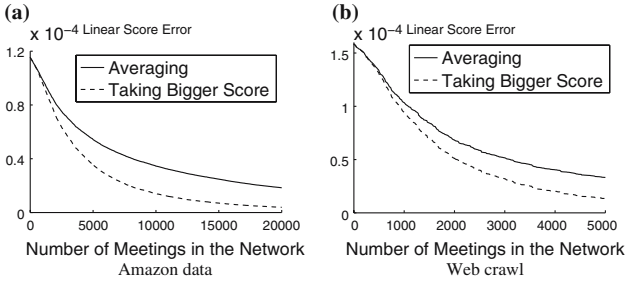

Fig. 8 Comparison of the methods for combining the score lists

Figures 9 and 10 present the performance comparison between the two peer selection strategies, with the pre-meet-ings phase and without the pre-meetings phase, where peers are chosen at random, for the Amazon data and the Web crawl, respectively.

We can see that during the first meetings both approaches perform similarly, but as peers discover, through the pre-meet-ings, the most promising peers, the number of meetings need-ed for a good approximation to the global PR scores is reduced. For instance, in the Amazon data, to make the footrule distance drop below 0.1 we needed a total of 16,180 meetings without the pre-meetings phase. With the pre-meetings phase this number was reduced to 7,340. In the Web crawl setup, for a footrule distance of 0.05, the number of meetings was reduced from 9,930 to 5,670. It is clear that the peer selection strategy plays a big role not only on

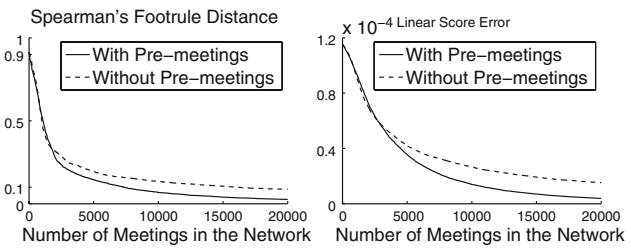

Fig. 9 Comparison of peer selection strategies for the Amazon data

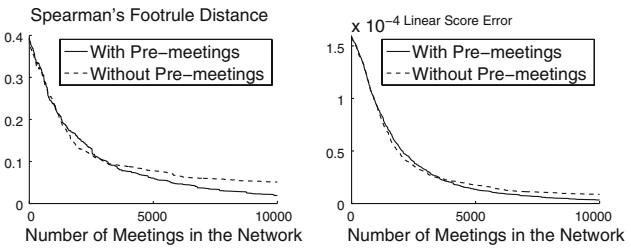

Fig. 10 Comparison of peer selection strategies for the Web crawl

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 marginally 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 0.4 was around 507 MB with the pre-meetings phase, compared to the 621 MB transmitted when meetings were performed at random—a reduction of almost 20%. In the Web crawl, the decrease in the amount of bytes transmitted, for a footrule distance of 0.2, was about 30%, from 5.34 to 3.69 GB. 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.

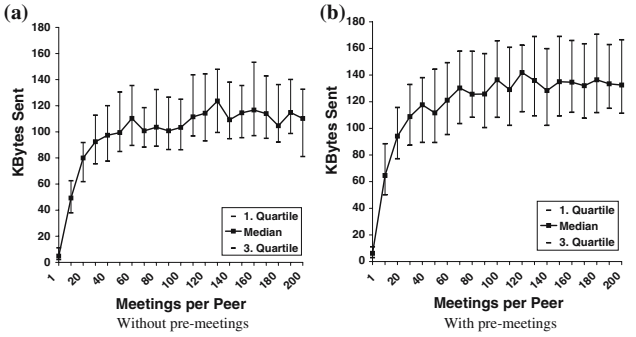

Fig. 11 Message size (in kB) for the Amazon data setup

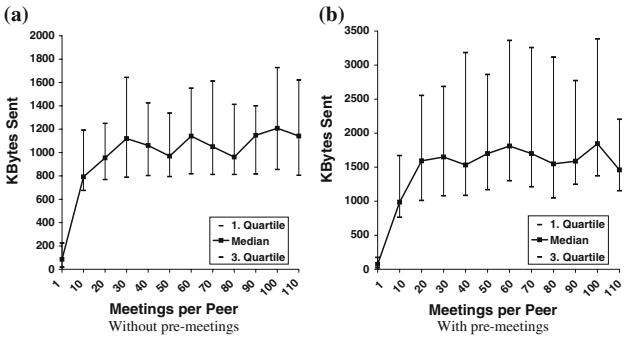

Fig. 12 Message size (in kB) for the Web crawl setup

7.3 Effects of misestimating the global number of pages

Here we again measured Spearman's footrule distance and the Linear Score Error at the top-1,000 for the JXP and PR global rankings for the both collections. Additionally we have computed the L1 norm for the JXP ranking vector and the cosine between the full ranking vectors of JXP and PR. Since the scores are normalized, the L1 norm for the global PR vector is 1. Figures 13 and 14 show the results for values of X equals to N, $10N$, $5N$ and $0.5N$.

We can see that Spearman's footrule distance and the cosine measure are not affected by the different choices of X, which is an indication that the JXP scores are affected only by a rescaling factor, and that the ranking order is not altered.

7.4 Scalability

We have also studied the scalability of the JXP algorithm, by varying the number of peers on the network. We have tested networks with 100, 200 and 500 peers. For these settings, the total number of Web pages distributed over peers was 21121, 35096, and 59496 for the Amazon data and 52049, 79298, and 123896 for the Web crawl. The pre-meetings strategy was used, and the size of the friends and candidates lists did

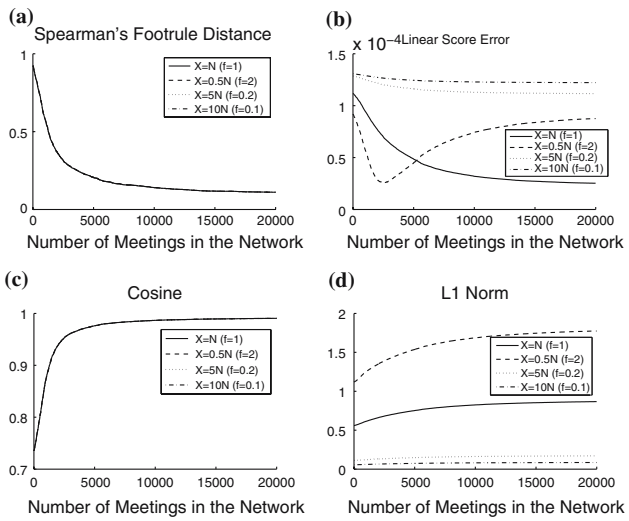

Fig. 13 Experimental results for X equals to N, $0.5N$, $5N$ and $10N$ for the Amazon data

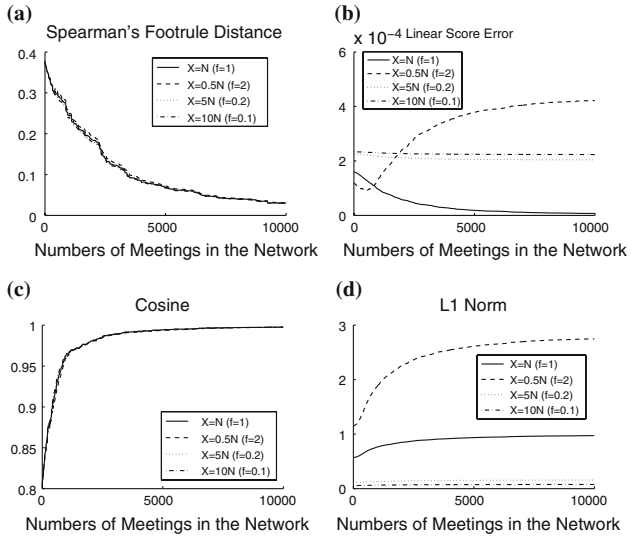

Fig. 14 Experimental results for X equals to N, $0.5N$, $5N$ and $10N$ for the Web crawl

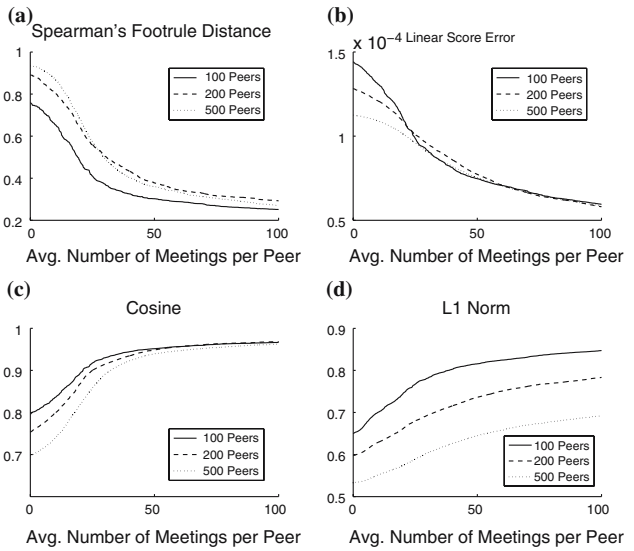

Fig. 15 Performance with different numbers of peers for the Amazon data

not change with the number of peers in the network. Results are shown in Figs. 15 and 16. For a better comparison of the results, the x-axis shows the average number of meetings per peer, i.e., total number of meetings divided by the number of peers in the network.

The results show that, as the numbers of peers in the network increases, even though the total number of meetings increases, the average number of meetings a peer has to perform for the same approximation quality does not vary that much.

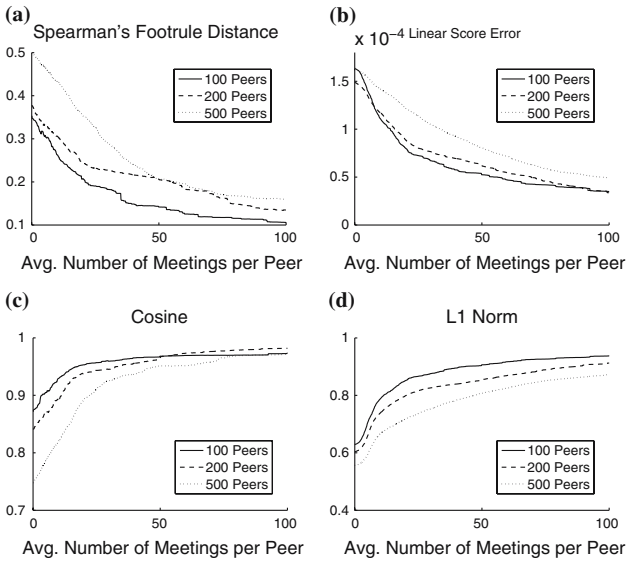

Fig. 16 Performance with different numbers of peers for the Web crawl

7.5 TrustJXP performance

In these experiments we created malicious peers by "cloning" some of the already existing peers as follows. The number of malicious peers was varied in the experiments (see below). Each malicious peer copied the local graph from one of the 100 good peers. It cheats at all meetings by either reporting a higher score for all or some of its local pages, or by permuting the scores among its pages. Malicious peers perform meetings and local PageRank computations like any normal peer. The difference is that, when asked by another peer for its scores list, a malicious peers will lie about the scores of its local pages. In the experiments, peers do not change their behavior during the TrustJXP computation; for example, if a peer chooses to permute its scores for the first meeting, it will do so for all subsequent meetings and it will apply always the same permutation.

Starting from the setup with 100 honest peers we first introduced 10 cheating peers. Each of these 10 peers uses one of the possible attacks by uniformly random choice (i.e., with each one of three types of misbehavior having probability 1/3 to be chosen by a dishonest peer):

- Some peers always report local JXP scores that are twice as high as their true values for all of their local pages.
- Some peers always report these falsely boosted scores for only half of their local pages (drawn randomly but used consistently throughout all meetings).
- Some peers always permute their scores list.

We kept this setup of mixed behaviors, and increased the number of dishonest peers from 10 to 50. The results of this

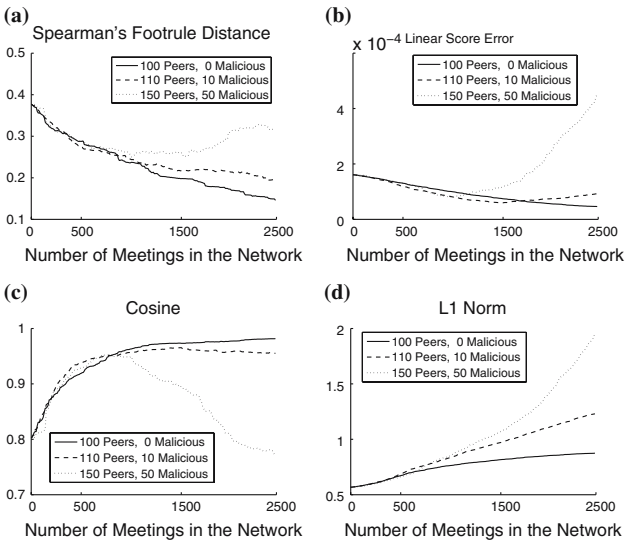

Fig. 17 Impact of Malicious peers with original JXP

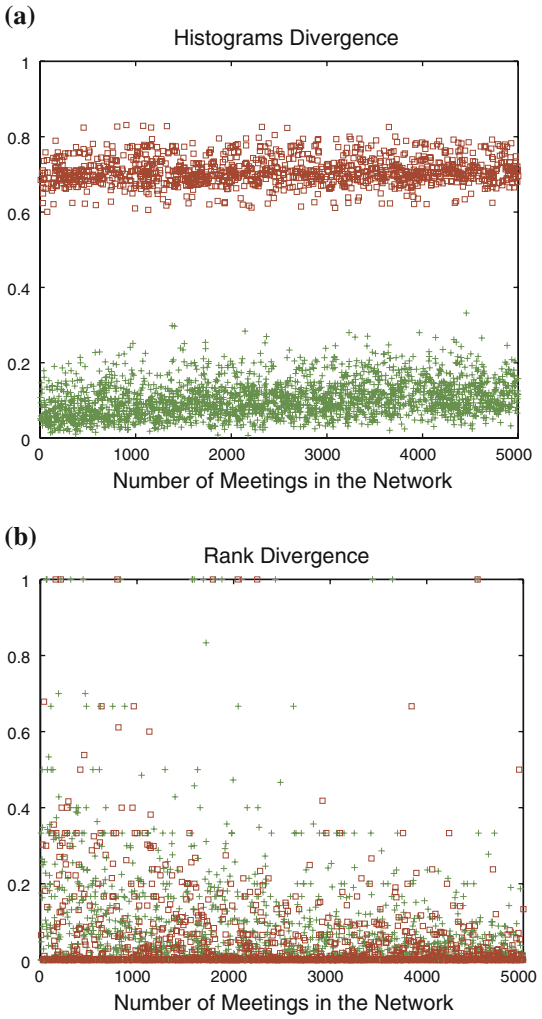

Fig. 18 Histograms and rank divergence with increased-scores attack

experiment, when using standard JXP without countermeasures to cheating, are shown in Fig. 17. We clearly see that, with the introduction of malicious peers and without any trust mechanism, the JXP scores do no longer converge to the true global PageRank values. The mathematical analysis of the JXP algorithm given in Sect. 5 proved that the JXP scores are upper-bounded by the true PageRank scores. With malicious peers reporting scores that are higher than the true ones, there is no bound for the scores. This effect can escalate: it distorts the world-node score and the transition probabilities from the world node to the local pages, and can even lead to a negative transition probability for the word node's self loop. At this point, scores start becoming undefined; this is the point where the linear-error, cosine, and L1-norm curves stop.

We proceeded by testing our trust model, measuring both histograms divergence and rank divergence for the overlapping pages. We again introduced 50 cheating peers, but now all peers performed the same type of attack. Figures 18 and 19 show the Hellinger Distance and the Tolerant Kendall's Tau for the case where cheating peers report scores five times higher than the true ones, and for the case where peers permute their scores, respectively. In these graphics a green (or light grey) "plus" symbol denotes that an honest peer met another honest peer, and a red (or black) "square" symbol means that an honest peer met a cheating peer. Meetings performed among dishonest peers are not shown for the sake of clarity. The results confirm our hypothesis that comparing histograms can be an effective indicator of cheating behavior with increased scores. We can also see that, when scores are permuted, the histogram approach does no longer work, and the rank divergence provides a better indication of such malicious behavior.

Finally, we repeated this experiment with 50 malicious peers, and used our TrustJXP method for computing local scores. The histograms and rank divergence, as well as the final TrustJXP scores are shown in Fig. 20, and the performance of the TrustJXP algorithm is illustrated in Fig. 21.

For measuring how effective a trust model can be at all, we simulated the "ideal" case, with an oracle-based defense mechanism that knows the status of each peer (honest vs. cheating) and thus can detect bad behavior with 100 percent accuracy. The results for the ideal case are also shown in Fig. 21. Of course, the ideal behavior cannot be implemented in practice, but it serves as a gold-standard yardstick for our methods. We can see that, for most of the metrics, our TrustJXP method is fairly close to the ideal case in terms of detecting and compensating malicious peers.

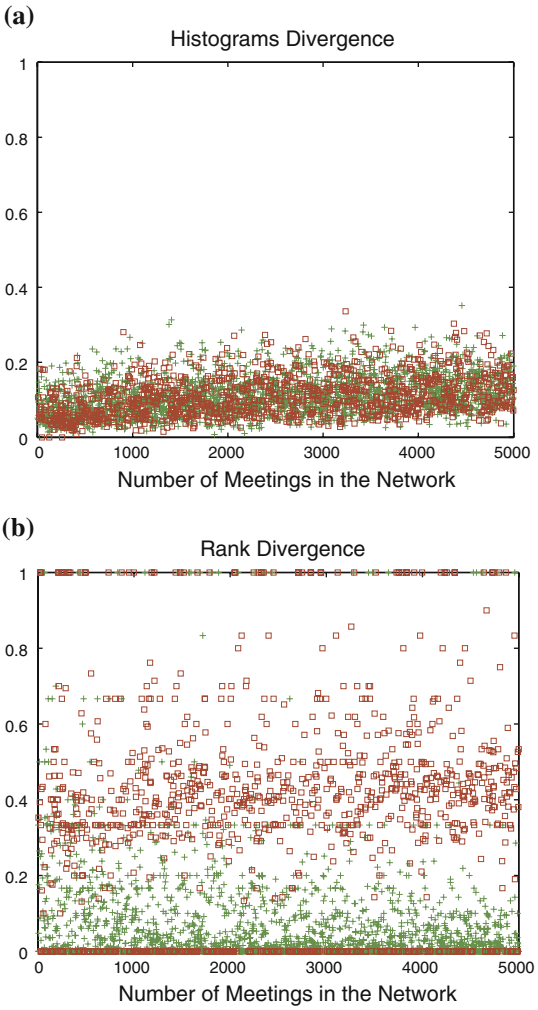

Fig. 19 Histograms and rank divergence with permuted-scores attack

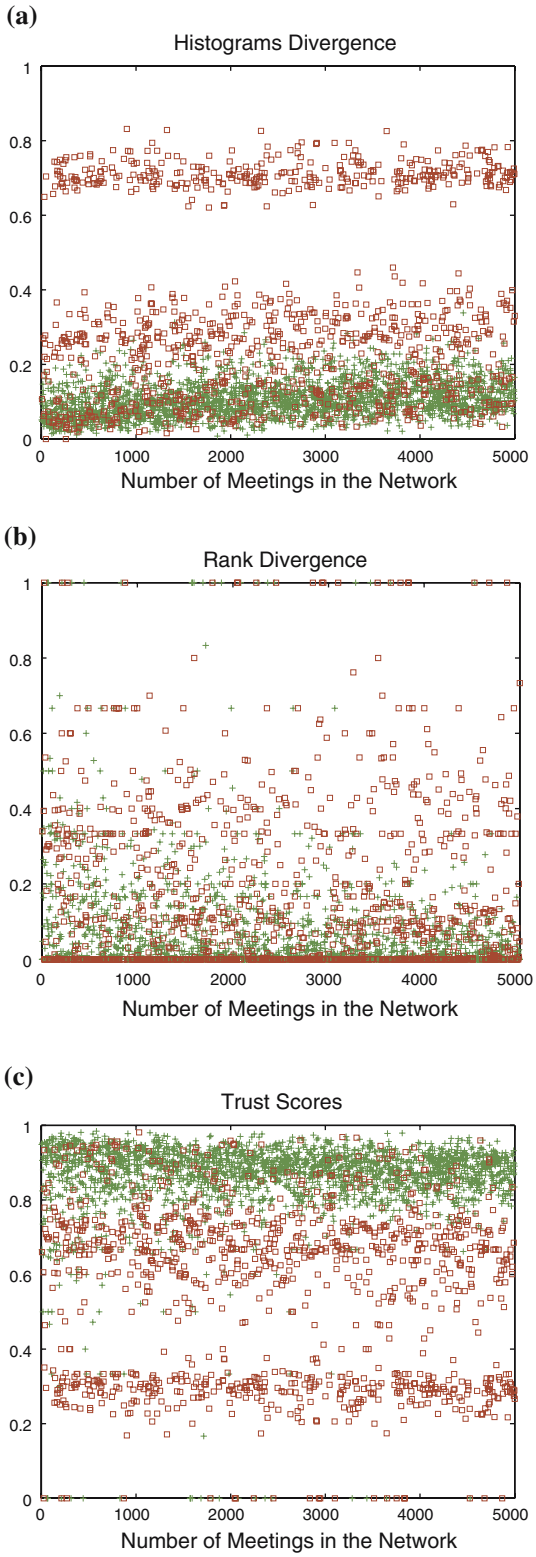

Fig. 20 Histograms divergence, rank divergence and trust scores. Random forms of attack

7.6 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 [8,10]. 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 category sets by splitting each set into four 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.

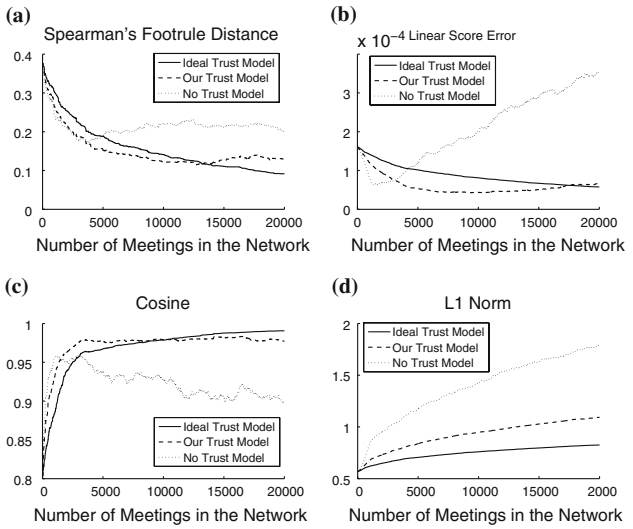

Fig. 21 Impact of Malicious peers with TrustJXP method

Then we ran 15 queries that are typical for popular Web search requests [13], 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 [13] and have been intensively used in prior literature on link analysis. We manually 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%.

Table 2 Precision at top-10 for the Web Collection

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**

8 Conclusions

This paper has presented the JXP and TrustJXP methods for robust and efficient computation of approximate PageRank scores in a fully decentralized manner that blends well with P2P networks where peers are autonomous and the local data collections of peers may overlap. Our methods are versatile and could be easily adapted to computer other kinds of authority and trust measures that are based on Eigenvectors in some form of social network. A salient property of JXP is its scalability: regardless of how large the network becomes, the storage and computational costs per peer are limited by the (order of the) resource commitments that the peer has made for hosting its local data collection and graph fragment anyway. Also, the messaging costs for peer meetings are very small. JXP scores provably converge to global PageRank values. The convergence speed depends on the network size, but we have developed smart peer-meeting strategies that accelerate convergence as demonstrated in our experiments. Finally, for robustness to peers that may cheat when exchanging information in a peer meeting, our TrustJXP extensions provide effective means to combat various forms of misbehavior.

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.

Our experiments, with two different datasets and systematic variation of setups, have confirmed the anticipated properties of JXP: convergence to global PR values and low computational costs. They also showed that the various optimizations that were developed in Sect. 4 pay off by accelerating convergence and reducing networking as well as local processing costs. Despite the relatively small scale

of our experiments (caused by limitations of our experimental machinery), we are very confident that, by the design of our methods and their salient properties, JXP will exhibit very good scalability in real-life large-scale environments as well. We are working on setting up larger-scale experiments in our lab and also with external collaboration partners over wide-area Internet connections. The experimental results for TrustJXP in the presence of dishonest peers are preliminary at this point but very encouraging. We can effectively detect and counter a certain degree of cheating, under several attacker models. But we surely realize also the limitations of the approach so far: if the fraction of misbehaving peers becomes excessive, no countermeasure whatsoever will be able to compensate the adverse effects of bad peers. Similarly, our attacker models themselves are limited at this point, and we will strive for a better, principled understanding of the ways that peers can attempt to cheat and manipulate others in such kinds of Web and social graph structures embedded in P2P networks.

For our future work, in addition to deeper studies of misbehavior and further improvement and extension of TrustJXP, we plan to explore how JXP performs in scenarios with very high dynamics. This includes both data and network dynamics. The global Web graph evolves at a high rate while we are carrying out P2P computations of the JXP style. The challenge here is twofold: on one hand, we want capture recent trends in this process and derive global authority measures that are as fresh as possible; on the other hand, we need to make sure that we are not misled by a moving target and our result are not distorted by the fast evolution of the underlying data. As for network dynamics, the main problem to tackle is the so-called churn phenomenon: peers join and leave the network at a high rate and without giving notice. We want to safeguard JXP against these rapid and unpredictable forms of variability.

References

1. Aberer, K.: P-grid: a self-organizing access structure for p2p information systems. In: CoopIS, pp. 179–194 (2001)
2. Aberer, K., Wu, J.: A framework for decentralized ranking in web information retrieval. In: APWeb, pp. 213–226 (2003)
3. Abiteboul, S., Preda, M., Cobena, G.: Adaptive on-line page importance computation. In: WWW Conference, pp. 280–290. ACM Press (2003)
4. Abrams, Z., McGrew, R., Plotkin, S.: A non-manipulable trust system based on eigentrust. SIGecom Exch. **5**, 21–30 (2005)
5. Becchetti, L., Castillo, C., Donato, D., Fazzone, A.: A comparison of sampling techniques for web characterization. In: LinkKDD (2006)
6. Becchetti, L., Castillo, C., Donato, D., Leonardi, S., Baeza-Yates, R.: Using rank propagation and probabilistic counting for link-based spam detection. In: WebKDD. ACM Press, Pennsylvania (2006)
7. Benczúr, A.A., Csalogány, K., Sarlós, T., Uher, M.: Spamrank: Â fully automatic link spam detection. In: AIRWeb. Chiba (2005)
8. Bender, M., Michel, S., Parreira, J.X., Crecelius, T.: P2P web search: make it light, make it fly. In: CIDR 07, p. 6. Asilomar (2007)
9. Bender, M., Michel, S., Triantafillou, P., Weikum, G.: Global document frequency estimation in peer-to-peer web search. In: WebDB 2006, Chicago (2006)
10. Bender, M., Michel, S., Triantafillou, P., Weikum, G., Zimmer, C.: Minerva: collaborative p2p search. In: VLDB, pp. 1263–1266 (2005)
11. Berkhin, P.: A survey on pagerank computing. Internet Math. **2**(1), 73–120 (2005)
12. Bloom, B.H.: Space/time trade-offs in hash coding with allowable errors. Commun. ACM **13**(7), 422–426 (1970)
13. Borodin, A., Roberts, G.O., Rosenthal, J.S., Tsaparas, P.: Link analysis ranking: algorithms, theory, and experiments. ACM TOIT **5**(1), 231–297 (2005)
14. Brin, S., Page, L.: The anatomy of a large-scale hypertextual web search engine. In: WWW7, pp. 107–117 (1998)
15. Broder, A.: On the resemblance and containment of documents. In: SEQUENCES, p. 21. IEEE Computer Society, Washington (1997)
16. Broder, A.Z., Charikar, M., Frieze, A.M., Mitzenmacher, M.: Min-wise independent permutations. J. Comput. System Sci. **60**(3), 630–659 (2000)
17. Broder, A.Z., Lempel, R., Maghoul, F., Pedersen, J.O.: Efficient pagerank approximation via graph aggregation. Inf. Retr. **9**(2), 123–138 (2006)
18. Canright, G., Engo-Monsen, K., Jelasity, M.: Efficient and robust fully distributed power method with an application to link analysis. Tech. Rep. UBLCS-2005-17, University of Bologna, Department of Computer Science, Bologna (2005)
19. Chakrabarti, S.: Mining the Web: Discovering Knowledge from Hypertext Data. Morgan-Kauffman (2002)
20. Chen, Y.Y., Gan, Q., Suel, T.: Local methods for estimating pagerank values. In: CIKM, pp. 381–389. ACM Press (2004)
21. Chien, S., Dwork, C., Kumar, R., Simon, D.R., Sivakumar, D.: Link evolution: analysis and algorithm. Internet Math. **1**(3), 277–304 (2004)
22. Cho, G., Meyer, C.: Markov chain sensitivity measured by mean first passage times. Tech. rep., NCSU Technical Report #112242-0199 (1999)
23. Courtois, P.: Decomposability: Queueing and Computer System Applications. Academic, New York (1977)
24. Cuenca-Acuna, F.M., Peery, C., Martin, R.P., Nguyen, T.D.: Planetp: using gossiping to build content addressable peer-to-peer information sharing communities. In: HPDC, p. 236. IEEE Computer Society, Washington, (2003)
25. Dill, S., Kumar, R., Mccurley, K.S., Rajagopalan, S., Sivakumar, D., Tomkins, A.: Self-similarity in the web. ACM Trans. Inter. Tech. **2**(3), 205–223 (2002)
26. Drost, I., Scheffer, T.: Thwarting the nigritude ultramarine: learning to identify link spam. In: ECML, Lecture Notes in Artificial Intelligence, vol. 3720, pp. 233–243. Porto (2005)
27. Fagin, R., Kumar, R., Sivakumar, D.: Comparing top k lists. In: SIAM Discrete Algorithms (2003)
28. Fan, L., Cao, P., Almeida, J., Broder, A.Z.: Summary cache: a scalable wide-area web cache sharing protocol. IEEE/ACM TON **8**(3), 281–293 (2000)
29. Flajolet, P., Martin, G.N.: Probabilistic counting algorithms for data base applications. J. Comput. System Sci. **31**(2), 182–209 (1985)
30. Guha, R., Kumar, R., Raghavan, P., Tomkins, A.: Propagation of trust and distrust. In: WWW, pp. 403–412. ACM Press, New York (2004)
31. Gyöngyi, Z., Berkhin, P., Garcia-Molina, H., Pedersen, J.: Link spam detection based on mass estimation. In: VLDB, pp. 439–450 (2006)

32. Gyöngyi, Z., Garcia-Molina, H.: Web spam taxonomy. In: AIRWeb (2005)
33. Gyöngyi, Z., Molina, H.G., Pedersen, J.: Combating web spam with trustrank. In: VLDB, pp. 576–587. Morgan Kaufmann, Toronto (2004)
34. Jelasity, M., Montresor, A., Babaoglu, O.: Gossip-based aggregation in large dynamic networks. ACM Trans. Comput. Syst. **23**(3), 219–252 (2005)
35. Kalnis, P., Ng, W.S., Ooi, B.C., Tan, K.L.: Answering similarity queries in peer-to-peer networks. Inf. Syst. **31**(1), 57–72 (2006)
36. Kamvar, S., Haveliwala, T., Manning, C., Golub, G.: Exploiting the block structure of the web for computing pagerank. Technical report, Stanford University (2003)
37. Kamvar, S.D., Schlosser, M.T., Garcia-Molina, H.: The eigentrust algorithm for reputation management in p2p networks. In: WWW, pp. 640–651. ACM Press, New York (2003)
38. Kemeny, J.G., Snell, J.L.: Finite Markov Chains. Van Nostrand, Toronto – New York (1963)
39. Kempe, D., Dobra, A., Gehrke, J.: Gossip-based computation of aggregate information. In: FOCS, p. 482. IEEE Computer Society, Washington (2003)
40. Kempe, D., McSherry, F.: A decentralized algorithm for spectral analysis. In: STOC, pp. 561–568. ACM Press, New York (2004)
41. Kleinberg, J.M.: Authoritative sources in a hyperlinked environment. J. ACM **46**(5), 604–632 (1999)
42. Lamport, L.: Specifying Systems: The TLA+ Language and Tools for Hardware and Software Engineers. Addison-Wesley Longman, Boston (2002)
43. Langville, A., Meyer, C.: Updating the stationary vector of an irreducible markov chain with an eye on google's pagerank. In: SIMAX (2005)
44. Langville, A.N., Meyer, C.D.: Deeper inside pagerank. Internet Math. **1**(3), 335–400 (2004)
45. Langville, A.N., Meyer, C.D.: Deeper inside pagerank. Internet Math. **1**(3), 335–400 (2004)
46. Le Cam, L.: Asymptotic Methods in Statistical Theory. Springer, New York (1986)
47. Liben-Nowell, D., Balakrishnan, H., Karger, D.: Analysis of the evolution of peer-to-peer systems. In: PODC, pp. 233–242. ACM Press, New York (2002)
48. Marti, S., Garcia-Molina, H.: Taxonomy of trust: categorizing p2p reputation systems. Comput. Netw. **50**(4), 472–484 (2006)
49. Meyer, C.: Matrix Analysis and Applied Linear Algebra. SIAM, Philadelphia (2000)
50. Ntarmos, N., Triantafillou, P.: Seal: Managing accesses and data in peer-to-peer sharing networks. Peer-to-Peer Comput. **00**, 116–123 (2004)
51. Parreira, J.X., Donato, D., Michel, S., Weikum, G.: Efficient and decentralized pagerank approximation in a peer-to-peer web search network. In: VLDB. Seoul (2006)
52. Parreira, J.X., Weikum, G.: Jxp: Global authority scores in a p2p network. In: WebDB, pp. 31–36 (2005)
53. Podnar, I., Rajman, M., Luu, T., Klemm, F., Aberer, K.: Scalable peer-to-peer web retrieval with highly discriminative keys. In: ICDE (2007)
54. Ratnasamy, S., Francis, P., Handley, M., Karp, R.M., Shenker, S.: A scalable content-addressable network. In: SIGCOMM, pp. 161–172 (2001)
55. 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)
56. 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)
57. Shi, S., Yu, J., Yang, G., Wang, D.: Distributed page ranking in structured p2p networks. In: ICPP (2003)
58. Sizov, S., Theobald, M., Siersdorfer, S., Weikum, G., Graupmann, J., Biwer, M., Zimmer, P.: The bingo! system for information portal generation and expert web search. In: CIDR (2003)
59. Stakhanova, N., Basu, S., Wong, J., Stakhanov, O.: Trust framework for p2p networks using peer-profile based anomaly technique. In: ICDCS Workshops, pp. 203–209 (2005)
60. Steinmetz, R., Wehrle, K. (eds.): Peer-to-peer systems and applications, Lecture Notes in Computer Science, vol. 3485. Springer (2005)
61. Stewart, W.: Introduction to the Numerical Solution of Markov Chains. Princeton University Press, Princeton (1994)
62. Stoica, I., Morris, R., Karger, D., Kaashoek, M.F., Balakrishnan, H.: Chord: A scalable peer-to-peer lookup service for internet applications. In: SIGCOMM, pp. 149–160. ACM Press, New York (2001)
63. Suel, T., Mathur, C., Wen Wu, J., Zhang, J., Delis, A., Kharrazi, M., Long, X., Shanmugasundaram, K.: Odissea: a peer-to-peer architecture for scalable web search and information retrieval. In: WWW (2003)
64. Wang, Y., DeWitt, D.J.: Computing pagerank in a distributed internet search system. In: VLDB (2004)
65. Wu, B., Goel, V., Davison, B.D.: Propagating trust and distrust to demote web spam. In: Workshop on Models of Trust for the Web. Edinburgh, Scotland (2006)
66. Wu, J., Aberer, K.: Using a layered markov model for distributed Web ranking computation. In: ICDCS (2005)
67. Xiong, L., Liu, L.: Peertrust: supporting reputation-based trust for peer-to-peer electronic communities. IEEE Trans. on Knowl. Data Eng. **16**(7), 843–857 (2004)