

Knowing Where to Search: Personalized Search Strategies for Peers in P2P Networks

Paul - Alexandru Chirita, Wolfgang Nejdl and Oana Scurtu

L3S and University of Hannover

Deutscher Pavillon Expo Plaza 1

30539 Hannover, Germany

{chirita,nejdl,scurtu}@l3s.de

July 1, 2004

Abstract

Optimizing and focusing search and results ranking in P2P networks becomes more and more important with the increasing size of these networks. Even though a few approaches have already started to investigate the computation of PageRank-like values in P2P environments, none so far has investigated how personalization could be added to it. This paper tackles the problem of distributedly computing Personalized PageRank values in such a distributed environment and presents an algorithm which uses them to optimize and focus search in the P2P network. The paper also discusses how these algorithms improve current distributed search in power law networks and gives some simulation results.

1 Introduction

P2P networks are powerful distributed infrastructures capable of handling enormous amounts of resources while keeping an organized and balanced structure. As the amount of data data is continuously increasing, faster and more focussed search algorithms for such environments and more personalized approaches to rank results are needed. So far, however, only few distributed searching algorithms which also incorporate page or document ranks have been developed [11], none of them addressing *personalization*. Additionally, rank information can also improve the search focus and the selection of peers to forward queries to, as well as computing / exploiting reputation information. The algorithms discussed in this paper address these issues. Furthermore, they exploit the power law distribution observable in many current P2P networks and scale well with increasing numbers of peers in the network.

The contributions of this paper include (1) an algorithm for computing Personalized PageRank in a distributed fashion, (2) new approaches to searching P2P networks using (personalized) page ranks, and (3) experimental results of running these algorithms in a simulated "power law"-based P2P environment.

We start with a short introduction of the algorithms related to our research in section 2. Section 3 introduces the distributed computation of personalized page ranks and section 4 explains how to use this algorithm to perform focussed search in a P2P network. Experimental results are presented in section 5. Section 6 concludes with discussion of further work.

2 Background and Previous Work

2.1 Web Graph and PageRank

In the paper we will use a notation similar to [7]. $G = (V, E)$ represents the *Web graph*, where V is the set of all Web pages and E is the set of directed edges $\langle p, q \rangle$. E contains an edge $\langle p, q \rangle$ iff a page p links to page q . $I(p)$ denotes the set of in-neighbors (pages pointing to p) and $O(p)$ the set of out-neighbors (pages pointed to by p). For each in-neighbor we use $I_i(p)$ ($1 \leq i \leq |I(p)|$), the same applies to each out-neighbor. We refer $v(p)$ to denote the p -th component of \mathbf{v} . We will typeset vectors in boldface and scalars (e.g., $v(p)$) in normal font.

Let A be the adjacency matrix corresponding to the Web graph G with $A_{ij} = \frac{1}{|O(j)|}$ if page j links to page i and $A_{ij} = 0$ otherwise.

PageRank is an algorithm for computing a Web page score based on the graph inferred from the link structure of the Web. It is based on the idea that “a page has high rank if the sum of the ranks of its backlinks is high”. Given a page p , the PageRank formula is:

$$PR(p) = (1 - c) \sum_{q \in I_p} \frac{PR(q)}{\|O(q)\|} + cE(p) \quad (1)$$

The dumping factor $c < 1$ (usually 0.15) is necessary to guarantee convergence and to limit the effect of rank sinks [2]. Intuitively, a random surfer will follow an outgoing link from the current page with probability $(1 - c)$ and will get bored and select a random page with probability c .

2.2 Personalized PageRank

Description. [7] is the most recent investigation towards personalized page ranks. One Personalized PageRank Vector (PPV) is computed for each user. The personalization aspect of this algorithm stems from a *set of hubs* (H), each user having to select her *preferred pages* from it. PPVs can be expressed as a linear combination of basis vectors (PPVs for preference vectors with a single non-zero entry corresponding to each of the pages from P , the preference set), which could be selected from the precomputed basis hub vectors, one for each page from H . To avoid the massive storage resources the basis hub vectors would use, they are decomposed into partial vectors (which encode the part unique to each page, computed at run-time) and the hub skeleton (which captures the interrelationships among hub vectors, stored off-line).

Algorithm. In the first part of the paper, the authors present three different algorithms for computing basis vectors: “Basic Dynamic Programming”, “Selective Expansion” and “Repeated Squaring”. In the second part, specializations of these algorithms are combined into a general algorithm for computing PPVs, as depicted below.

Algorithm 1. Personalized PageRank in a centralized fashion.

Let $D[p]$ be the approximation of a basis vector corresponding to page p , and $E[p]$ the error of its computation.

1.(Selective Expansion) Compute the partial vectors using

$Q_0(p) = V$ and $Q_k(p) = V \setminus H$, for $k > 0$, in the formulas below:

$$\mathbf{D}_{k+1}[p] = \mathbf{D}_k[p] + \sum_{q \in Q_k(p)} c \cdot E_k[p](q) \mathbf{x}_q$$

$$\mathbf{E}_{k+1}[p] = \mathbf{E}_k[p] - \sum_{q \in Q_k(p)} E_k[p](q) \mathbf{x}_q + \sum_{q \in Q_k(p)} \frac{1-c}{|O(q)|} \sum_{i=1}^{|O(q)|} E_k[p](q) \mathbf{x}_{O_i(q)}$$

Under this choice, $D_k[p] + c * E_k[p]$ will converge to $\mathbf{r}_p - \mathbf{r}_p^H$, the partial vector corresponding to p .

2.(Repeated squaring) Having the results from the first step as input, one can now compute the hubs skeleton ($r_p(H)$). This is represented by the final $D[p]$ vectors calculated using $Q_k(p) = H$ into:

$$\mathbf{D}_{2k}[p] = \mathbf{D}_k[p] + \sum_{q \in Q_k(p)} E_k[p](q) * D_k[q]$$

$$\mathbf{E}_{2k}[p] = \mathbf{E}_k[p] - \sum_{q \in Q_k(p)} E_k[p](q) \mathbf{x}_q + \sum_{q \in Q_k(p)} E_k[p](q) E_k[q]$$

3. Let $u = \alpha_1 p_1 + \dots + \alpha_z p_z$ be a preferred vector,

where p_i are from H and i is between 1 and z , and let:

$$r_u(h) = \sum_{i=1}^z \alpha_i (r_{p_i}(h) - c * x_{p_i}(h)), \quad h \in H$$

which can be computed from the hubs skeleton.

The PPV v for u can then be constructed as:

$$v = \sum_{i=1}^z \alpha_i (r_{p_i} - r_{p_i}^H) + \frac{1}{c} \sum_{h \in H} r_u(h)_{>0} r_u(h) * [(r_h - r_h^H) - c * x_h]$$

For a more in-depth view of this process, we refer the reader to [7].

2.3 Distributed PageRank

There have been several recent approaches to computing distributedly the PageRank scores. Their motivation arises from the continuously increasing size of the Web graph [11, 12] and from the need of reputation models in P2P networks [15, 8]. Faster updates through incremental computation of ranks are also described. However, there is no algorithm which computes *personalized* page ranks in a distributed fashion.

In a P2P environment, each peer is computing the ranks of its documents, sending the intermediate results to all documents to which its documents are linked (out-links) [11]. The overall performance can be increased as in [12], where selected peers handle the computation for bigger groups of documents. High overall speed is obtained reducing the communication between peers to the minimum, provided that they are not overloaded by their local computations.

3 Computing Personalized PageRank in P2P Networks

In the distributed algorithm for computing PPR (Personalized PageRank), we will use the same set of hubs for all peers, but each peer will have its own preference set $P \subset H$. Interaction will be allowed only with direct neighbors and with peers from H .

We present a simplified version of the distributed algorithm first, and then discuss its extension for a real P2P network. In the initial version, each peer will cover *one* document to be ranked. We divide the algorithm in three parts, as presented in section 2.2: one focused mostly on peers from $V \setminus H$ (Algorithms 3.1.1 and 3.1.2), one on peers from H (Algorithm 3.2) and the final one putting everything together (Algorithm 3.3).

3.1 Partial Vectors

This part of the algorithm has one special initialization step and several succeeding steps. Even though its focus is on peers from $V \setminus H$, peers $p \in H$ will also gather their components of \mathbf{D} and \mathbf{E} . In the first step, each peer $q \in V$ will compute $E[p](q)$. As peers $p \in H$ are known, each peer $q \in V$ can set its initial values $D_0[p](q)$ and $E_0[p](q)$ by itself, as below:

$$D_0[p](q) = \begin{cases} c & , q \in H \\ 0 & , otherwise \end{cases} \quad (2)$$

$$E_0[p](q) = T_0[p](q) = \begin{cases} 1 & , q \in H \\ 0 & , otherwise \end{cases} \quad (3)$$

Non-hub peers. After this initialization, the following operations will be executed in parallel by each peer $q \in V \setminus H$ for each $p \in H$:

Algorithm 3.1.1. Distributed computation of partial vectors by peers in $V \setminus H$.

- 1: Send $\frac{1-c}{|\mathbf{O}(q)|} \cdot T_k[p](q)$ to all out-going neighbors (peers from which q has downloaded files), including those from H .
 - 2: Wait from all in-going neighbors (peers which have downloaded files from q) their $T_k[p](*)$ at this step (k)
 - 3: After all $T_k[p](*)$ values have been received, compute $T_{k+1}[p](q)$ as:

$$T_{k+1}[p](q) = \sum_{v \in \mathbf{I}(q)} T_k[p](v)$$
 - 4: Compute:

$$D_{k+1}[p](q) = D_k[p](q) + c \cdot T_k[p](q)$$

$$N$$
 being the number of steps we apply the Selective Expansion algorithm.
 - 5: If there are more iterations left, go to 1
 - 6: Each peer $q \in V \setminus H$ computes $(r_p - r_p^H)(q) = D_N[p](q) + c \cdot T_N[p](q)$, its component of the partial vector corresponding to p .
-

Theorem 3.1. In the distributed version of the Selective Expansion algorithm, for $p \in H$ each peer $q \in V \setminus H$ has to compute:

$$T_{k+1}[p](q) = \sum_{v \in \mathbf{I}(q)} \frac{1-c}{|\mathbf{O}(v)|} \cdot E_k[p](v) \quad (4)$$

Proof.

$$\begin{aligned} E_{k+1}[p](q) &= \left(E_k[p] - \sum_{v \in V \setminus H} E_k[p](v) \cdot x_v \right) (q) + \\ &\quad + \left(\sum_{v \in V \setminus H} \frac{1-c}{|\mathbf{O}(v)|} \sum_{i=1}^{|\mathbf{O}(v)|} E_k[p](v) \cdot x_{O_i(v)} \right) (q) = \\ &= E_k[p](q) - (E_k[p](q) \cdot x_q) (q) + \\ &\quad + \left(\sum_{v \in \mathbf{I}(q)} \frac{1-c}{|\mathbf{O}(v)|} \sum_{i=1}^{|\mathbf{O}(v)|} E_k[p](v) \cdot x_{O_i(v)} \right) (q) = \\ &= \left(\sum_{v \in \mathbf{I}(q)} \frac{1-c}{|\mathbf{O}(v)|} E_k[p](v) \cdot x_q \right) (q) = \\ &= \sum_{v \in \mathbf{I}(q)} \frac{1-c}{|\mathbf{O}(v)|} \cdot E_k[p](v) \end{aligned}$$

□

Hub peers. In the special first step, a peer $p \in H$ will send $\frac{1-c}{|\mathbf{O}(q)|} \cdot T_kp$ to its all out-going neighbors (peers from which p has downloaded files), including those from H . After that, it will execute the following operations:

Algorithm 3.1.2. Distributed computation of partial vectors by peers in H .

- 1: Wait from all in-going neighbors (peers which have downloaded files from p) their $T_k[p](*)$ at this step (k)
 - 2: After all $T_k[p](*)$ values have been received, compute $T_{k+1}p$ as $T_{k+1}p = \sum_{v \in \mathbf{I}(q)} T_k[p](v)$
 - 3: If there are more iterations left, go to 1
 - 4: Compute: $Ep = \sum_{k=1}^N T_kp$
 - 5: Set D_Np to c
 - 6: Each peer $p \in H$ computes $(r_p - r_p^H)(p)$, its component of its partial vector.
-

Algorithms 3.1.1 and 3.1.2 could be reduced to PageRank, and they would therefore converge also in the presence of loops in G (see [7, 9] for details).

3.2 Hub Skeleton

In the second phase of the algorithm, each peer $p \in H$ has to calculate its hub skeleton ($r_p(\mathbf{H})$) using as input the results from the previous stage. The result is stored in the values $\mathbf{D}_{2k}[p]$ obtained after the last iteration of the following operations, performed by each $p \in H$:

Algorithm 3.2. Distributed computation of hub skeleton (only in H).

- 1: Calculate $\mathbf{D}_{2k}[\mathbf{p}]$ and $\mathbf{E}_{2k}[\mathbf{p}]$, using the same formulas as the centralized version.
- 2: Multicast the results to all other peers $q \in H$, possibly using a minimum spanning tree for that.
- 3: If there are more iterations left, go to 1.
- 4: Every hub peer broadcasts its $\mathbf{D}_{2N}[\mathbf{p}]$ sub-vector (only the components regarding pages from H).

As this step refers to hub-peers only, the computation of $\mathbf{D}_{2k}[\mathbf{p}]$ and $\mathbf{E}_{2k}[\mathbf{p}]$ can consider *only* the components regarding pages from H .

3.3 PPV

As the $\mathbf{D}_{2N}[\mathbf{p}]$ sub-vectors ($p \in H$) have been broadcast, *any* peer $v \in V$ can now determine its $\mathbf{r}_v(\mathbf{P})$ locally, using the original formula (see section 2.2).

Any peer $v \in V$ can also calculate its partial PPV containing its rank and the ranks of its neighbors from its own point of view. If we denote NB the set of neighbors of v , it must do the following:

Algorithm 3.3. Computation of the Personalized PageRank Vector.

- 1: Request the components of $\mathbf{r}_p - \mathbf{r}_p^H$ for all $p \in H$ from all neighbors (peers from NB).
 - 2: Compute the components of the PPV using the original formula (see section 2.2).
-

3.4 Discussion

In the algorithm described above, one peer deals only with one value, i.e. with the rank of one document. Usually a peer would manage several documents, in which case it would only have to perform the same computations for each of these documents (i.e. act as being a set of peers, each peer handling one document). Considering that most of the links are intra-domain links, the overall performance will be much higher, because there will be less network traffic and more local computations.

A second issue is about what should a peer compute. Rather than computing its rank and the ranks of its neighbors, it could compute only its rank or the ranks of all the other peers (from its own point of view). However, in a P2P environment it is less useful and more complicated to put together the ranks of all peers in the network.

4 Using Personalized PageRanks for P2P Search

4.1 Description

One can introduce page ranks into several of the existing P2P searching techniques. We present an approach similar to [1], and focus the forwarding of queries by using rank values we have computed in a distributed fashion:

Algorithm 4. Searching P2P Networks using Distributed Personalized PageRanks.

- 1: Upon issuing a query, a peer p will send it to its neighbor having the highest rank from its perspective (or the neighbor containing the highest ranked page). It will attach to the query its own ID (or IP address), the number of results desired and a TTL (time to live).
- 2: Upon initial receiving of a query, a peer q will:
 - 2.1: Add its own ID (or IP address) to the query and decrease the TTL.
 - 2.2: Check if the local page matches the query. If yes, then send it to p and decrease the number of results desired.

- 2.3: If the number of results desired is more than zero, select the neighbor with the highest rank whose ID is not already added to the query (highest ranked unvisited neighbor) and forward the query to it.
 - 2.4: If there is no unvisited neighbor, send a special message back to the peer from which the query was obtained. Put in the message the latest instance of the query (i.e. with latest list of visited peers and most recent number of desired results).
- 3: Upon receiving a query back from a neighboring peer, go to step (2.3).
-

4.2 Analysis of Other Search Algorithms

Searching P2P networks depends on the architecture used (e.g. super-peer based or pure P2P networks), on the expected availability, etc. Although techniques as those of Chord [13] or CAN [10] guarantee location of content within a limited number of hops, they exhibit a tight control over the network (e.g. topology, placement, etc.). Similar to [16] our work is focused on file-sharing systems with less tight constraints, such as Gnutella [6] or Freenet [5].

[16] explores three techniques, called "Iterative Deepening", "Directed BFS" and "Local Indices". The first one is about initiating multiple breadth-first searches with successively larger depth limits until either enough answers to the query have been obtained or the maximum depth limit has been reached. It does not necessarily visit the best peers which can answer the query, but it generates lower network traffic than the other approaches. "Directed BFS" sends the query only to the "best" neighbors of a peer, which are computed by judging the previous experiences achieved when querying. In this paper, we use PPR values, which do not include only the local experience, but also the experiences of all other peers. The "Local Indices" technique deals with building descriptions of the content of all neighboring peers, thus allowing fast identification of peers able to answer the query. This is orthogonal to our method, but we intend to investigate how they could be combined to provide better results.

Two other search techniques are discussed in [1], namely one in which queried neighbors are randomly selected, and another in which the unvisited neighbor with the highest degree is queried. Although eventually the former method gravitates around high-degree nodes, the latter one reaches them faster. However, using PageRank scores as criterion for determining the queried node should provide even better results than simply using the node-degree.

5 Experiments

5.1 Distributed Personalized PageRank

We tested our algorithms on simulated P2P networks which exhibit a power-law degree distribution. For the distributed ranking computation, we generated a graph with 215,000 nodes, an out-going link structure with the power law exponent $\gamma = 2.7$ and an in-going link structure with $\gamma = 2.1$. The set of hubs contained 150 (hub) peers, while the preference set of each ordinary peer contained 30 hub peers (out of the 150 from above) randomly selected. Results are summarized in tables 1 and 2.

	Max. Size	Avg. Size
Hub Peers, All Data	33089622	32996141
Hub Peers, $\neq 0$ Data	14372218	6273058

Table 1: Data transferred by hub peers (nb. of real values sent)

	Max. Size	Avg. Size
Non Hub Peers, All Data	2905500	48093
Non Hub Peers, $\neq 0$ Data	82824	103

Table 2: Data transferred by non-hub peers (nb. of real values sent)

We distinguish in our statistics between ordinary peers and hub peers, especially because they are doing different operations and the amount of data sent differs quite a lot from one category to the other. During the simulation, we discovered many peers often send the value "0" through the network (as a correct intermediate value). We decided also to count these "special" 0 values in order to verify whether a further modification of the algorithm which would avoid sending them is indeed useful or not. The current experimental results indicate a positive answer.

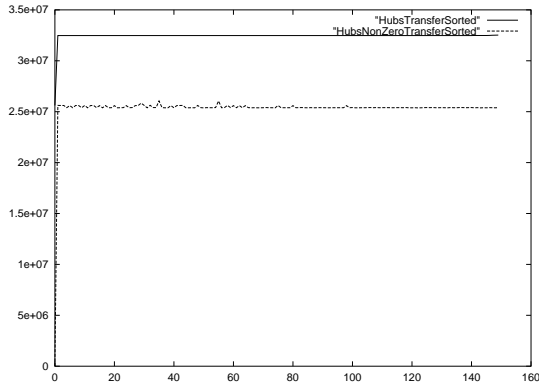


Figure 1: Data transfered by hub peers (nb. of real values sent)

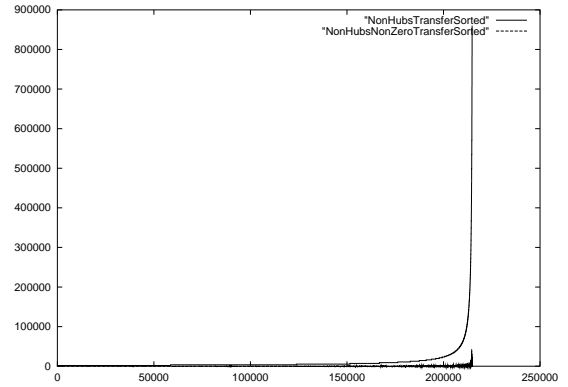


Figure 2: Data transfered by non-hub peers (nb. of real values sent)

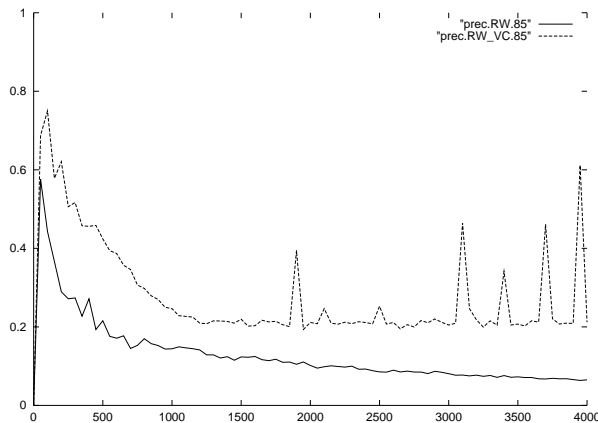


Figure 3: Precision of random walk search, with and without visiting same peers again

The hub peers generate significantly more traffic because of the second phase of the computation (algorithm 3.2), in which they need to multicast their intermediate results to all other hubs in the network. However, there is usually a small number of hub peers compared to the size of the network, and they also control more resources (e.g. bigger bandwidth or processing power). Finally, we distinguish the reduced communications of ordinary peers, which means that for the majority of the network, the computation of PPRs would need only small bandwidth.

Figures 1 and 2 present the traffic generated by each peer in the network. As we are dealing with a power-law network, one can observe the power-law distribution of data size among the ordinary peers. This is because the more neighbors a peer has, the more data it needs to exchange. Bigger (and therefore better) versions of all our figures can be found in the extended version of this paper [3].

5.2 Search in P2P Networks

We analyzed our algorithm against several similar approaches:

1. Best-neighbor search (as in [1]: at each step, the query is forwarded to the unvisited neighbor having the biggest number of neighbors).
2. PageRank search: forward the query to the neighbor with the highest PageRank (i.e. without personalization).

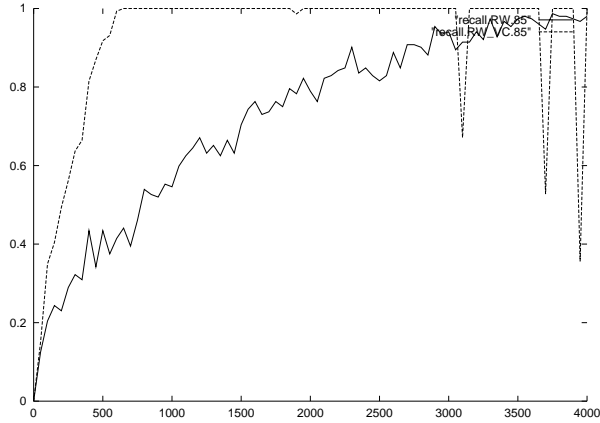


Figure 4: Recall of random walk search, with and without visiting same peers again

The latter solution is new. Even though there are several articles building PageRank in a distributed manner, they do not use our search approach.

We use a simulated power-law network with the same exponents as in the previous sub-section, but with 10,000 peers. Each of them has one document with a size varying from 100 to 5000 again after a power-law distribution. The random words we used also exhibit a probability of appearance in a document which follows a power-law.

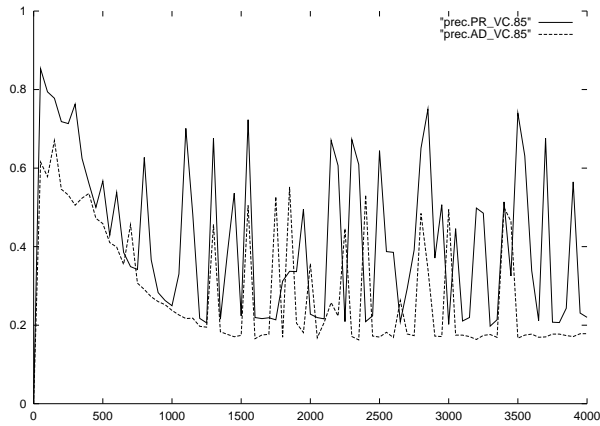


Figure 5: Precision of Best Neighbor and PageRank-based search

In order to evaluate the results, we used precision and recall with respect to the top PageRank results (see equation 5), as well as an estimation of the necessary TTL to get some specific results. When not specified, all our graphs contain the TTL on the x-axis.

$$Precision = \frac{\text{Nb. of Top 200 matches found}}{\text{Total nb. of matches found}} \quad Recall = \frac{\text{Nb. of Top 200 matches found}}{\text{Nb. of Top 200 matches}} \quad (5)$$

We started with some experiments inspired from the walk of a random surfer [9], i.e. in which a peer forwards a query using some algorithm to determine the next target with probability $c < 1$, and to any randomly selected peer with probability $1 - c$. Note that jumping to a peer in a P2P network is not straightforward, but we included these initial searches for comparison purposes.

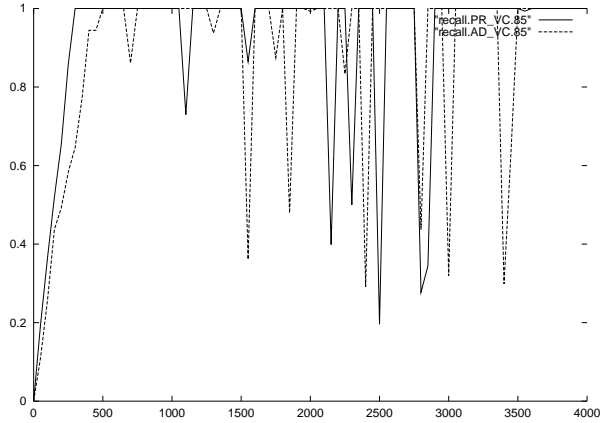


Figure 6: Recall of Best Neighbor and PageRank-based search

In the first two experiments (figures 3 and 4), the peer randomly selects one of his neighbors with probability $c = 0.85$ and jumps to any peer with probability 0.15. The difference in precision and recall is given by the fact that in RW_VC we ensured that the query will not visit the same peer again unless absolutely necessary (there is no other neighbor to visit). The small irregularities in the graphs are given by the random jumps which sometimes lead to "poor" peers.

We continued in the same manner, testing a neighbor-based and a PageRank-based search (figures 5 and 6). As predicted, the latter approach performs slightly better. Although selecting the neighbor with the largest number of neighbors finds best peers relatively fast, it usually cannot find them faster than when the actual best neighbor is selected.

We also tested these last two experiments without random jumps. We found both better precision and better recall when using PageRank to select the neighbor where to forward the query, especially with a $TTL > 150$. Again we observed some small irregularities, but they are normal considering the fact that all documents were randomly generated at the beginning of each run (in order not to have them stored in a file).

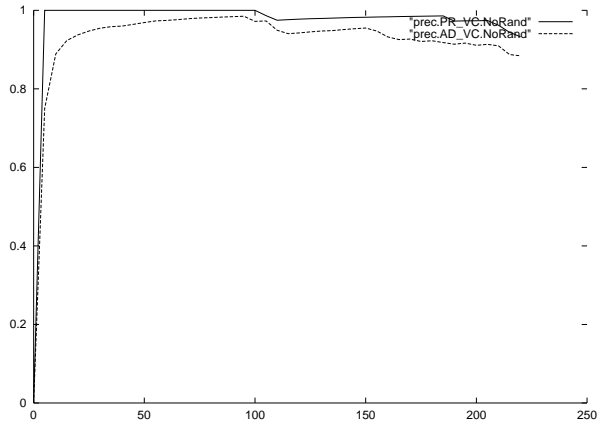


Figure 7: Precision of Best Neighbor and PageRank-based search

Finally, we wanted to simulate the PPR search. As all our documents are randomly generated, we were forced to use the following heuristic: in PPR the documents from my preference set will contain the keywords of my interest with a higher probability ($prob' = \alpha \cdot prob$) than in PageRank, because top PPR documents will be on

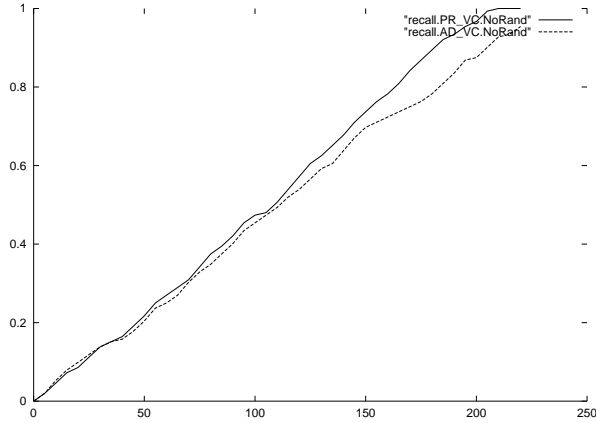


Figure 8: Recall of Best Neighbor and PageRank-based search

the same topic as my topic of interest. However, we cannot estimate how much higher this will be (considering our experience from [4], α varies from one preference set/topic to another). Therefore, we decided to test using $\alpha = \{2, 3\}$, being sure that $\alpha > 1$. The results are depicted in figures 9 and 10. Even though PPR (PR_VC_2, PR_VC_3) performed better than PageRank, increasing α over 2 does not seem to bring much improvements.

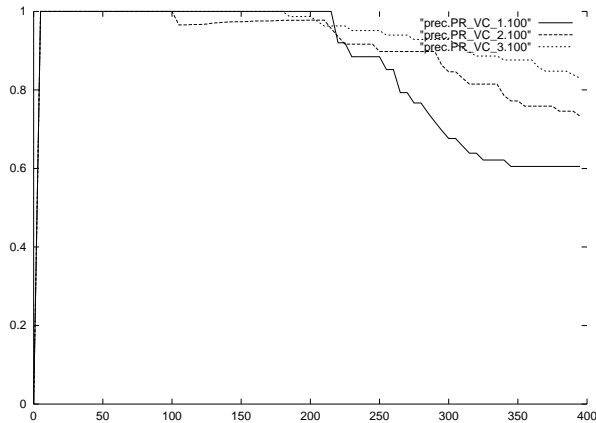


Figure 9: Precision of simulations of the PPR

In the last experiment we measured the TTL needed to get the top 30 (figure 11) and top 50 (figure 11) PageRank documents using different algorithms: neighbor-based search (AD), PageRank, Personalized PageRank $\alpha = 2$ (PR_2) and Personalized PageRank $\alpha = 3$ (PR_3). On the x-axis there are different words, "10000" being the most popular of them and "0" the least popular. We can see that especially for unpopular words, PPR-based search finds the best results much faster (e.g. getting the top 30 results querying for the most popular word needs TTL 38 for both AD and PPR, while for say word 6000, we need TTL 200 for AD, but only 100 for PPR).

6 Conclusions and Further Work

In this paper we have presented the first algorithm which computes personalized rank values in a P2P network. We used this algorithm to perform neighbor-based searches which improves selection of neighbors simply based

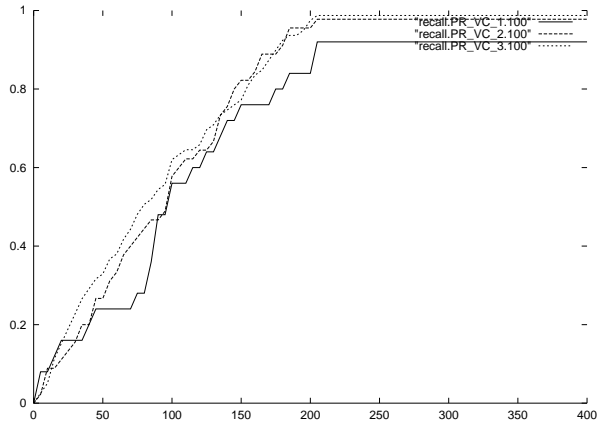


Figure 10: Recall of simulations of the PPR

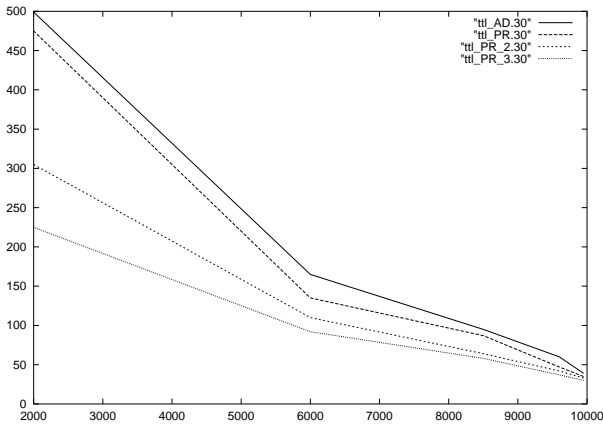


Figure 11: TTL needed to get top 30 PageRank pages

on their node-degree (as done in [1]).

In the future we want to extend this work by investigating a directed BFS approach, in which the query is sent to the top-N neighbors rather than only to the best one. Additionally, we are thinking about further improvements to our ranking algorithm and about integrating it into a top-k P2P search [14].

References

- [1] L.A. Adamic, R. Lukose, A. Puniyani, and B. Huberman. Search in power-law networks. *Phys. Rev., E* 64, 046135, 2001.
- [2] Sergey Brin, Rajeev Motwani, Lawrence Page, and Terry Winograd. What can you do with a web in your pocket? *Data Engineering Bulletin*, 21(2):37–47, 1998.
- [3] Paul-Alexandru Chirita, Wolfgang Nejdl, and Oana Scurtu. Knowing where to search: Personalized search strategies for peers in p2p networks. Technical report, L3S and University of Hannover, 2004. <http://www.l3s.de/chirita/pros/pros.html>.

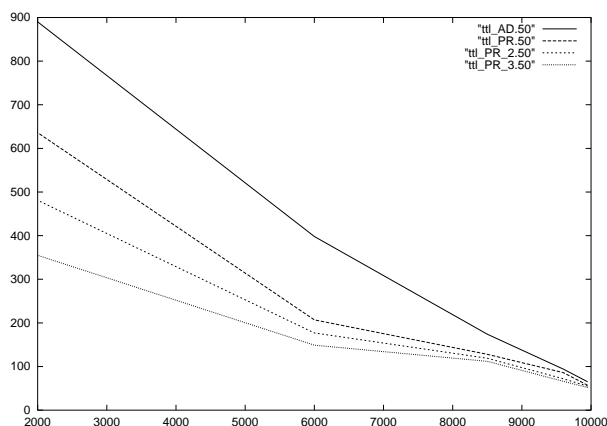


Figure 12: TTL needed to get top 50 PageRank pages

- [4] Paul-Alexandru Chirita, Daniel Olmedilla, and Wolfgang Nejdl. Pros: A personalized ranking platform for web search. In *Proceedings of the International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems*, Aug 2004.
- [5] The freenet project: <http://freenet.sourceforge.net/>.
- [6] Gnutella web page: <http://www.gnutella.com/>.
- [7] G. Jeh and J. Widom. Scaling personalized web search. In *Proceedings of the 12th International World Wide Web Conference*, 2003.
- [8] S. Kamvar, M. Schlosser, and H. Garcia-Molina. The eigentrust algorithm for reputation management in p2p networks. In *Proceedings of the 12th International World Wide Web Conference*, 2003.
- [9] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford University, 1998.
- [10] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Schenker. A scalable content-addressable network. In *Proceedings of the 2001 conference on applications, technologies, architectures, and protocols for computer communications*, pages 161–172. ACM Press, 2001.
- [11] K. Sankaralingam, S. Sethumadhavan, and J. C. Browne. Distributed pagerank for p2p systems. In *Proceedings of the 12th IEEE International Symposium on High Performance Distributed Computing*, 2003.
- [12] S. Shi, J. Yu, G. Yang, and D. Wang. Distributed page ranking in structured p2p networks. In *Proceedings of the 2003 International Conference on Parallel Processing*, 2003.
- [13] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, San Diego, USA, August 2001.
- [14] Uwe Thaden, Wolf Siberski, and Wolfgang Nejdl. Top-k query evaluation for schema-based p2p networks. Technical report, L3S and University of Hannover, 2004.
- [15] A. Yamamoto, D. Asahara, T. Ito, S. Tanaka, and T. Suda. Distributed pagerank: A distributed reputation model for open peer-to-peer networks. In *Proceedings of the 2004 Symposium on Applications and the Internet-Workshops*, 2004.
- [16] B. Yang and H. Garcia-Molina. Efficient search in p2p networks. In *Proceedings 22nd International Conference on Distributed Computing Systems*, Austria, 2002.