Distributed Calculation of PageRank Using Strongly Connected Components

Michael Brinkmeier

Institute for Technical and Theoretical Computer Science,
Technical University of Ilmenau, Germany
mbrinkme@tu-ilmenau.de

Abstract. We provide an approach to distribute the calculation of PageRank, by splitting the graph into its strongly connected components. As we prove, the global ranking may be calculated componentwise, as long as the rankings of pages directly linking to the current component are already known. Depending on the structure of the WWW, this approach approach may be used to calculate the ranking on several components in parallel, and allows to split the problem intio significantly small subproblems.

1 Introduction

The World Wide Web is one of the most rapidly developing and perhaps the largest source of information. Due to its de-central nature the access to the relevant and needed information becomes increasingly difficult. Due to the pure amount of content, more and more search engines, indexes and archives try to harvest the information implicit in the link structure to improve speed and quality of search. One of the tools in this field is the ranking of web pages according to their relevance. PageRank, one of the most prominent systems, was presented by Page, Brin et al. in [4, 5]. It is an essential part of Google's ranking scheme. Together with content based measures, this purely link based value is the basis of the order of search results produced by this widely used and accepted search engine.

Since the ranking seems to be quite successful (if considering the number of users and their confidence in the results), the theoretical properties of PageRank raise some interesting questions and may allow a significant speed-up of its calculation. Usually PageRank is viewed as a Markov chain (e.g. [5, 12, 8, 10, 3]), even though its original definition does not constitute one, as pointed out by several authors. But if sinks, i.e. nodes without outgoing edges, are removed or connected to all other nodes, one obtains a Markov chain producing the same ranking as the original definition [3]. More general, PageRank is usually calculated by iteratively multiplying a (ranking) vector to a form of *normalized adjacency matrix* of the graph. Standard results of linear algebra and numerical mathematics show, that this iteration converges to the principal eigenvector of the normalized adjacency matrix.

A. Bui et al. (Eds.): IICS 2005, LNCS 3908, pp. 29–40, 2006.

Experiments and theoretical results prove that only a small number of iterations (compared to the size of the WWW) are needed to obtain a good approximation [5, 3, 6]. But nonetheless, the size of the graph usually requires the use of external memory, significantly increasing the required time.

In the literature some approaches can be found, suggesting parallelized or distributed calculation of PageRank. In [9] T. Haveliwala suggests a way to do one step of the iteration blockwise, reducing the number of accesses to the external memory. But this approach still requires the execution of each step of the iteration on the whole graph.

In [12] Kamvar et al. suggested to use the natural structure of the WWW for a faster calculation of PageRank. The web is split into local subwebs (for example domains), which are ranked independently. Then in a second step the net of subwebs is ranked. The resulting local and global rankings are combined to obtain an approximation of PageRank, which in turn is used as a starting vector for the standard iteration, increasing the rate of convergence. A similar approach is used in [14] by Wang and DeWitt, but instead of using the combined vectors as starting vector for the iteration, they use the global rankings (or ServerRank, as they call it) to refine the local rankings.

The approach of distributing the calculation of PageRank presented in this paper is more in the tradition of [2]. There Avrachenkov and Litvak proof that the global PageRank may be calculated from the local PageRanks on the weakly connected components. This allows to iterate each component seperateley and then combine them to obtain the global PageRank. But unfortunately, as former experiments show [7], there exists one weak giant component containing about 91% of the vertices, reducing the size of the main problem by about 9%.

Similar to Arasu et al. in [1], we go a step further and prove that the iteration may be executed separately on each strong connected component, as long as we adhere to the structure of the interconnections of the strong components. In detail this means, that we have to calculate the rankings inside a strong component before the rankings inside other components, to which it links, directly or indirectly. As [7] and additional experiments indicate, this reduces the size of the largest subproblem to about 28% ([7]) or 46% (sec. 4), and the remaining 72%, resp. 54% consist of much smaller strong components.

2 PageRank and Strong Components

2.1 Notations

Let $D = (V, E)$ be a directed multigraph with vertex set V and edge set E. For each vertex v we denote the out-degree of v by $\mathrm{out}(v)$ and the in-degree by $\mathrm{in}(v)$. If there exists an edge from u to v we write $u \to v$, and $u \not\to v$ otherwise.

2.2 PageRank

In [4] and [5] Page, Brin et al. described an approach estimating the importance of a web page, based purely on the link structure of the world wide web. Their

proposed score *PageRank* was based on the assumption, that a document spreads its relevance equally to all documents it links to.

To 'generate' rank a fixed value $e(u)$, the *personalization value*, is given for each vertex u, and $(1-d)e(u)$ is added to the rank, resulting in:

$$\texttt{PageRank}(u) = d \sum_{v|v\to u} \frac{\texttt{PageRank}(v)}{\text{out}(v)} + (1-d)e(u).^{1} \tag{1}$$

Using the normalized link matrix of the web, i.e. the matrix $M = (m_{uv})$ with $m_{uv} = \frac{1}{\text{out}(u)}$ if there exists a link from u to v and 0 otherwise, this equation may be reformulated as the following linear system:

$$(I - dM^T)\texttt{PageRank} = (1-d)e, \tag{2}$$

with I the unit matrix. Under certain circumstances[2] the equation may be solved using the iteration

$$r_{i+1} = dM^T r_i + (1-d)e, \tag{3}$$

which corresponds to the iterative algorithm suggested by Page and Brin[3].

Translated to the underlying graph the iteration leads to the following equation and algorithm (1).

$$r_{i+1}(u) = d \sum_{v|v\to u} \frac{r_i(v)}{\text{out}(v)} + (1-d)e(u). \tag{4}$$

The starting vector r_0 can be chosen arbitrarily.

Page and Brin suggested an interpretation of the iteration in terms of a random surfer, who occasionally jumps or teleports to another page instead of following a link, leading to a Markov model. But unfortunately the normalized adjacency matrix M is not stochastic, since *sinks*, ie. vertices without outgoing edges, have columns summing to 0. Usually this problem is solved by adding virtual edges from each sink to all other vertices (including the sink), leading to a proper Markov model (see eg. [3, 9, 10]). Results of the theory of Markov models then ensure, that the iteration converges to a unique limit.

In [6] an alternative approach is used. There, PageRank is described as a power-series over the damping factor d, whose coefficients are probabilities of walks of a random surfer. In detail, it was proved that

$$\texttt{PageRank}(u) = \sum_{l=0}^{\infty} d^l (1-d) \sum_{v} a_l(v,u) e(v) \tag{5}$$

[1] In fact, in the original paper [4] the factor with which the personalization is multiplied, was given as d. But in later publications it was replaced by $(1-d)$. As we will see this influences the absolute values, but not on the ranking.

[2] The spectral radius of the matrix $I - dM^T$ has to be less than 1.

[3] Since they normalized the ranking vector, they could describe PageRank as an eigenvector of a specific matrix.

with

$$a_0(v,u) = \begin{cases} 1 & \text{if } v = u \\ 0 & \text{otherwise} \end{cases} \qquad \text{and} \qquad a_{l+1}(v,u) = \sum_{w|w\to u} \frac{a_l(v,w)}{\text{out}(w)}. \tag{6}$$

Using a slightly different notation, this allows another description as a sum over paths.

$$\texttt{PageRank}(u) = \sum_v e(v) \sum_{\pi:\, v \xrightarrow{*} u} P(\pi)(1-d)d^{l(\pi)}. \tag{7}$$

where $P(\pi)$ is defined inductively over the length l of the paths,

$$P(\pi) = \begin{cases} 0 & \text{if } l = 0 \text{ and } u \neq v \\ 1 & \text{if } l = 0 \text{ and } u = v \\ \frac{P(\pi')}{\text{out}(w)} & \text{if } l > 0 \text{ and } \pi\colon u \xrightarrow{\pi'} w \to v, \end{cases}$$

and π is the path π' from u to w followed by the edge (x, v).

This formulation of PageRank allows an alternative description in terms of a random surfer. Instead of choosing the start vertex uniformly, the personalization value $e(v)$ is used as probability for v[4]. at each step the surfer decides to either continue its walk with probability d, or to stop surfing (probability $1 - d$). If she decides to continue, two situations can occur. In the first case, the current vertex has outgoing links, and the surfer randomly chooses one of them to follow. In the second case, the vertex has no outgoing link. In this situation the surfer becomes 'annoyed' and stops surfing.

In this setting $\texttt{PageRank}(v)$ is the probability that the surfer ends his walk *voluntarily* in vertex v. The probability that the surfer becomes annoyed causes a loss of ranking, ie. $\sum \texttt{PageRank}(v) < 1$, but the resulting ranking is equivalent (cmp. [6, Prop. 2.9]).

2.3 Strong Components

An obvious consequence of equation 7 is the simple and intuitive fact, that the ranking of a vertex u is only influenced by that of another vertex v, if there exists a path from v to u, ie. if $a_l(v,u) \neq 0$ for some $l \geq \mathbb{N}$. In [6] this was already exploited for sinks and sources in the underlying graph. In this paper we go further.

A *strong component* C of a directed multigraph $D = (V, E)$ is a maximal subgraph of D, such that for two arbitrary vertices u and v of C there exists a path from u to v and vice versa. The next theorem states that the PageRank of an arbitrary vertex may be obtained by an iteration involving only the vertices of its strong component, assumed that the PageRanks of all vertices not in the component, but directly linking to vertices in the component, are known.

[4] We assume $e(v) \geq 0$ for each $v \in V$ and $\|e\|_1 = 1$.

Theorem 1. *Let $D = (V, E)$ be a directed multigraph, and C one of its strong components. Then for each $v \in V$ the sequence $r^{(i)}(v)$ with $r^{(0)}(v) = (1-d)e(v)$ and*

$$r^{(i+1)}(v) = d \sum_{u \in C | u \to v} \frac{r^{(i)}(u)}{\mathrm{out}(u)} + d \sum_{u \notin C | u \to v} \frac{\mathtt{PageRank}(u)}{\mathrm{out}(u)} + (1-d)e(v)$$

converges to $\mathtt{PageRank}(v)$.

Proof. If the sequence converges, the limit obviously has to be $\mathtt{PageRank}(u)$, because it has to satisfy the fixpoint condition (1), whose solution is unique (cf. [6, Thm 2.8]). Hence it remains to prove the convergence. This is done by comparison with the partial sums

$$\mathtt{PageRank}^{(i)}(v) = \sum_{l=0}^{i} (1-d)d^l \sum_{u \in V} a_l(u, v)e(u) \leq \mathtt{PageRank}(v),$$

which form an increasing sequence converging to $\mathtt{PageRank}(v)$.

Obviously, we have $\mathtt{PageRank}(v) \geq r^{(0)}(v) = \mathtt{PageRank}^{(0)}(v)$. Now assume

$$\mathtt{PageRank}(v) \geq r^{(i)}(v) \geq \mathtt{PageRank}^{(i)}(v)$$

for all $v \in C$ and $i \geq 0$. Then we have

$$\begin{aligned}
\mathtt{PageRank}(v) &= d \sum_{u | u \to v} \frac{\mathtt{PageRank}(u)}{\mathrm{out}(u)} + (1-d)e(u) \\
&= d \sum_{u \in C | u \to v} \frac{\mathtt{PageRank}(u)}{\mathrm{out}(u)} + d \sum_{u \notin C | u \to v} \frac{\mathtt{PageRank}(u)}{\mathrm{out}(u)} + (1-d)e(u) \\
&\geq d \sum_{u \in C | u \to v} \frac{r^{(i)}(u)}{\mathrm{out}(u)} + d \sum_{u \notin C | u \to v} \frac{\mathtt{PageRank}(u)}{\mathrm{out}(u)} + (1-d)e(v) \\
&= r^{(i+1)}(v) \\
&\geq d \sum_{u | u \to v} \frac{\mathtt{PageRank}^{(i)}(u)}{\mathrm{out}(u)} + (1-d)e(v) \\
&\geq \mathtt{PageRank}^{(i+1)}(v).
\end{aligned}$$

Since the partial sums form an increasing sequence, converging to $\mathtt{PageRank}$, the $r^{(i)}(v)$ have to have the same limit.

In the preceeding proof we neglected the question, wether the iteration is well-defined. But the answer to this question is quite simple and obvious. We iterate only on one strong component C of the graph. For each vertex v in this component, we require values for each predecessor u. But there are two types of predecessors. The first is itself a member of C and hence is included in the iteration. The second is not a member of C and hence its PageRank is assumed to be known.

At the first look, the result only allows us to iterate on a strong component C, if the exact PageRanks for the predecessors are known. But using the same estimations, one can prove the following result.

Theorem 2. *Let $D = (V, E)$ be a directed multigraph, and C one of the strong components of D and for each vertex u directly linking to at least one vertex in C, let $r(u)$ be an approximation of* $\texttt{PageRank}(u)$ *satisfying*

$$\texttt{PageRank}(u) \geq r(u) \geq \texttt{PageRank}^{(j)}(u)$$

for some $j \geq 0$. For $v \in C$ define $r^{(i)}(v)$ by $r^{(0)}(v) := (1-d)e(v)$ and

$$r^{(i+1)}(v) = d \sum_{u \in C | u \to v} \frac{r^{(i)}(u)}{\mathrm{out}(u)} + d \sum_{u \notin C | u \to v} \frac{r(u)}{\mathrm{out}(u)} + (1-d)e(v).$$

Then $\texttt{PageRank}() \geq r^{(j)}(v) \geq \texttt{PageRank}^{(j)}(v)$.

Proof. Use the same sequence of inequalities as for theorem 1, but replace $r(u)$ for $\texttt{PageRank}(u)$, where appropriate.

This result allows us to approximate PageRank componentwise, without losing precision, as long as the number of iterations is the same for each component. In fact, we may even obtain a better approximation, than by iterating the whole graph. Unfortunately, the quality of the approximations can not be guaranteed, if the common criteria for the termination of the iteration found in the literature is used. Usually the iteration is repeated until the L_1-norm of the difference between $r^{(i)}$ and $r^{(i+1)}$, ie. the sum $\sum_{v \in V} |r^{(i)}(v) - r^{(i+1)}(v)|$, is below a given threshold. This approach does not seem to be appropriate here, if this would lead to more iterations for a given strong component, than for its preceeding components, because the usage of the earlier calculated approximations of the rankings causes a bias for the newly calculated rankings. But if the number of iterations is independently fixed, we may even guarantee the quality of the global approximation, as shown in [6]. There it was proved that

$$\|\texttt{PageRank} - \texttt{PageRank}^{(i)}\|_1 = \sum_{v \in V} |\texttt{PageRank}(v) - \texttt{PageRank}^{(i)}(v)| \leq (1-d)d^i \|e\|_1,$$

if $(1-d)e$ is the initial ranking. Since $\texttt{PageRank}(v) \geq r^{(i)}(v) \geq \texttt{PageRank}^{(i)}(v)$, this implies

$$\|\texttt{PageRank} - r^{(i)}\|_1 \leq (1-d)d^i \|e\|_1.$$

If $\|e\|_1 = 1$ this implies that the error is less than ε after more than $\frac{\varepsilon}{1-d} \ln d$ iterations.

Since the rankings for all vertices $u \notin C$ linking to vertices in the strong component C are constant, their influence may be added to the initial ranking vector e. In this way, only the edges inside the strong component and the global out degrees have to be known. The edges into the component can be neglected.

Hence we may iterate on the induced subgraph with global out degrees (not with the outdegrees in the subgraph) using the initial rankings

$$e'(v) := e(v) + \frac{1}{1-d} \sum_{u \notin C | u \to v} \frac{r(u)}{\mathrm{out}(u)}.$$

The factor $1/(1-d)$ is required, because in the iteration the personalization vector is multiplied by $(1-d)$.

3 Distributing PageRank

The observations made above, allow us to calculate the PageRanks of all pages componentwise. Obviously, this has no effect when the graph is strongly connected. But as measurements of the Webgraph indicate, the World Wide Web consists of a lot of strong components, whose size follows a so-called Power-Law [7]. We will go further into detail about this in section 4.

Since the ranks of predecessors of a strong component C are required for the iteration, we have to ensure that these are known if we start the iteration on C. Hence we have to order the strong components appropriately. This step is discussed in the following.

Let $D = (V, E)$ be a directed multigraph. For each vertex v of D we denote the strong component of D containing v by $[v]$. The *strong component graph* $SC(D)$ of D is obtained by contracting each strong component into one vertex, deleting self-loops and merging parallel edges obtained by this procedure. In other words, the set of vertices of $SC(D)$ is the set $\{[v] \mid v \in V\}$ of strong components of D, and there exists an edge from component $[u]$ to $[v]$, if there exists an edge from u to v in D.

Obviously, the strong component graph $SC(D)$ is acyclic, since otherwise one strong component of D is distributed over several vertices of $SC(D)$ (at least the ones on the cycle). As a consequence, the calculation of PageRank can be distributed componentwise. We simply have to make sure, that before the rankings inside a strong component $[v]$ are calculated, the rankings in all preceeding strong components are already known.

First, the strong components may be computed using an algorithm of Tarjan [13], requiring $O(|V| + |M|)$ time, up to a constant the same time as an iteration step requires.

Following that, the rankings may be computed componentwise. We require a queue `Q` and one integer counter $c([v])$ for each strong component $[v]$. This counter is initially set to the indegree of $[v]$ in $SC(D)$, and counts the number of preceeding strong components not completed yet. If we guarantee, that $c([v]) = 0$ for every $[v]$ in the queue, we may simply extract one strong component from `Q` and calculate its rankings.

Obviously all *source components*, ie. those without incoming links, have $c([v])$ initially set to 0 and may be inserted into `Q`. If later, a strong component $[v]$ is fetched from the queue and its rankings are computed, the counters of all

Algorithm 1. Componentwise calculation of PageRank

```
forall [v] ∈ SC(D) do
    c([v]) ← in([v]);
    if in([v]) = 0 then  Q.append([v]);
end
while Q is not empty do
    [v] ← Q.get();
    Calculate the Rankings of vertices in [v];
    forall [u] with [v] → [u] do
        c([v]) ← c([v]) − 1;
        if c([v]) = 0 then Q.append([v]);
    end
end
```

successors can be decreased by one, and if one counter reaches 0 the strong component is inserted into `Q`. This leads to algorithm 1.

3.1 The System

If Algorithm 1 is executed on a single machine, the main gain is the reduction of the size of the graphs on which the iteration has to be done. This may enable the calculation in local memory, if the strong components are small enough. Fortunately, the topology of the WWW allows a more efficient calculation, as discussed in section 4, using more machines.

If a strong component is extracted from the queue, its rankings may be calculated completely independent from all other components in the queue at the same moment. Hence we may extract as many components as we have free machines, if possible. This allows us to speed up the complete calculation, if the topology of the strong component graph is good enough.

We assume that there exists a data storage system D storing the graph, the strong component graph and the rankings. Furthermore we assume, that this depot can be accessed by several clients $C_1, \ldots, C_n$ in parallel, without a significant loss of performance. In addition there exists a server S, storing the strong component graph and handling the queue.

The clients $C_1, \ldots, C_n$ may request an id of a strong component from the server, ready for calculation. The client retrieves the necessary data from D, ie. the strong component and the rankings of all vertices linking into this component, executes the calculation, stores the results in D and sends s message to S, indicating the completion of the calculation for the component. Following that, the server updates the queue and, if it is empty, stops the whole process.

4 Measurements

The measurements and experiments described in this section were conducted on the WebBase dataset from [16] constructed from the WebBase crawl of 2001 [15].

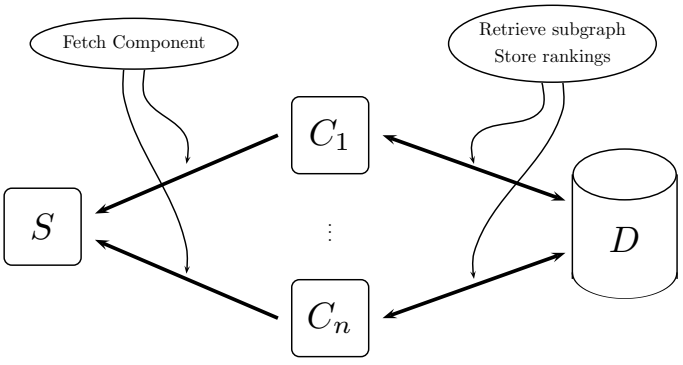

Fig. 1. A schematic sketch of the system for the distributed calculation of PageRank

Table 1. Basic numbers about the WebBase Dataset

Number of vertices	118 142 115
Number of edges	1 019 903 190
Number of strong components	41 126 852
Average size of strong components	~ 2.8726
Largest strong component	53 891 939
Second largest strong component	9 428
Third largest strong component	5 925
Number of strong components of size 1	39 843 421
Number of strong components of size 2	323 994
Number of strong components of size 3	154 786
Ratio of vertices in largest component	~ 0.456
Ratio of vertices in components of size ≤ 3	~ 0.341

Some basic numbers regarding the structure of the dataset are given in table 1. The distribution of the sizes clearly follows a power law (cmp. fig. 2).

As we can see, about 46% of the vertices form one giant strong component, while the second and third largest strong components contain 9428 and 5925 vertices. Furthermore about 34% are contained in tiny strong components of at most 3 vertices, resulting in an average size of strong components about 3.

These numbers indicate that the componentwise calculation of PageRank may significantly decrease the required time. Except for the giant component, every component may be held in the main memory of a standard computer, allowing a fast iteration. Due to the large number of very small components (≤ 3 vertices), PageRank of at least a third of the vertices may even be calculated without iteration, using a direct solution of a system of linear equations with at most 3 variables.

Assume that there exists a path of components in $SC(D)$ and that the iteration and the access to server and database for a component with n vertices

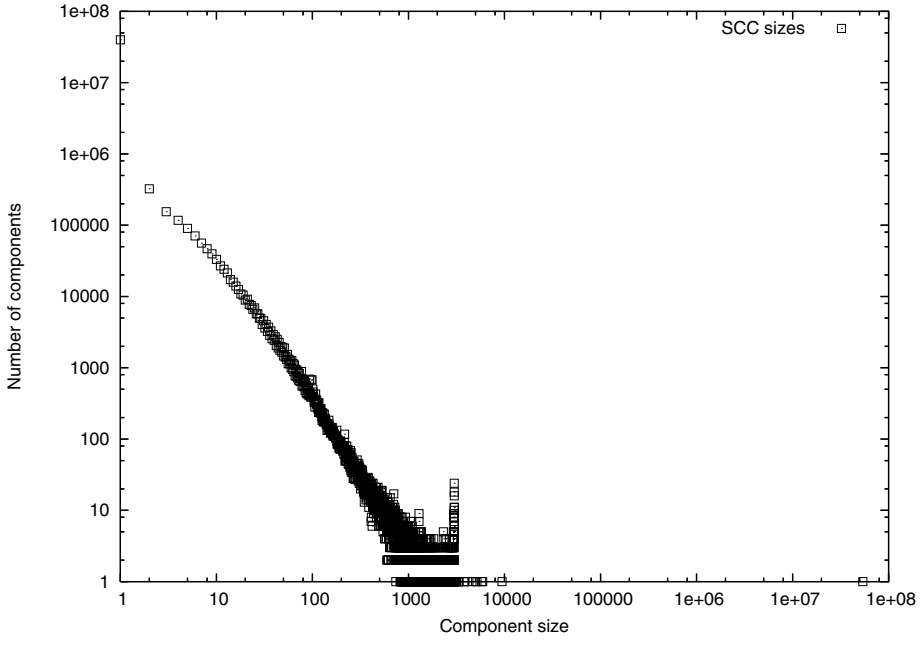

Fig. 2. The distribution of sizes of strong components in the WebGraph in logarithmic scale

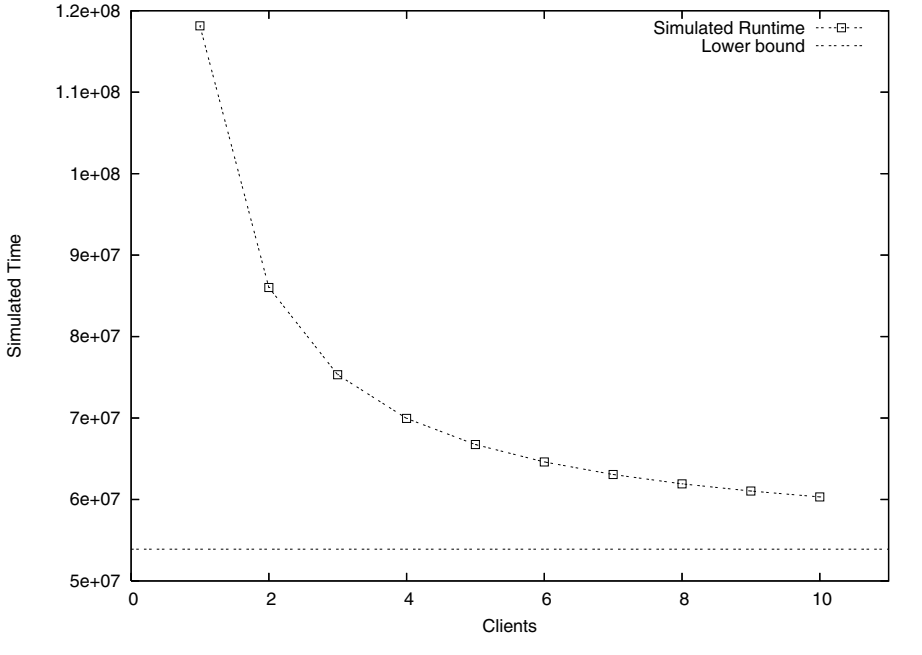

Fig. 3. The simulated time over the number of clients

takes about $O(n)$ time[5]. Since the components on this path has to be calculated subsequently, the time to calculate PageRank is bounded below by $\Omega(n)$. Hence we may use the maximal number of vertices in components on a path in $SC(D)$ from a source to a sink, as a lower bound for the time required for the calculation. This lower bound proved to be 53 903 795 vertices, which is only slightly more than the giant SCC of 53 891 939 vertices.

Using the above assumption, that the iteration takes linear time, the distributed calculation of PageRank was simulated to obtain estimations of the speedup. The server was implemented as a simple queue. The results for 1 to 10 clients are shown in figure 3. As the simulation demonstrates, the use of a small number of clients may reduce the required time significantly (2 clients to ca. 73%, 3 to ca. 64% and 4 to ca. 59%). But the speedup decreases with more clients.

4.1 Future Work

The results of the experiment seem promising, even though they indicate that only small number of clients are effective. But the conducted experiments have a major drawback. The measured "times" are not times used by real calculations of PageRank. They are estimations based solely on the topology of the strong component graph. Additional factors, like limited resources are not taken into account. These may be included in subsequent experiments.

In addition further extensions of the described system are possible. First of all, the rankings may easily be dynamically updated. As soon as a strong component changes, is split or several are composed, it may be inserted into the queue again and cause the recalculation of all succeeding components. Secondly, the calculation of the giant strong component may be distributed among several clients, reducing the resources required by single clients (but increasing the number of necessary communication and/or iterations). And last but not least, the clients may be weighted by their performance, restricting the size of the assigned components, and the extraction strategy of the queue may be varied to obtain better results.

References

[1] A. Arasu, J. Novak, A. Tomkins, and J. Tomlin. Pagerank computation and the structure of the web: Experiments and algorithms, 2001. citeseer.ist.psu.edu/arasu02pagerank.html.

[2] Konstantin Avrachenkov and Nelly Litvak. Decomposition of the google pagerank and optimal linking strategy. Technical Report RR-5101, Institut National de Recherche en Informatique et en Automatique, 2004.

[3] M. Bianchini, M. Gori, and F. Scarselli. Inside PageRank. *ACM Trans. Internet Tech.*, 5:92–128, 2005.

[5] In fact the required time for a constant number of iterations is about $O(n + m)$, where m is the number of edges. But since the average degree is small and can assumed to be constant, this bound can be viewed as $O(n)$.

[4] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual web-search engine. In *Proc. of the 7th World Wide Web Conference (WWW7)*, 1998.
[5] Sergey Brin, Lawrence Page, Rajeev Motwani, and Terry Winograd. The PageRank citation ranking: Bringing order to the web. Technical Report 1999-66, Stanford Digital Library Technologies Project, 1999. http://dbpubs.stanford.edu:8090/pub/1999-66.
[6] Michael Brinkmeier. Pagerank revisited. Technical report, Technical University Ilmenau, 2005. to appear.
[7] Andrei Z. Broder, Ravi Kumar, Farzin Maghoul, Prabhakar Raghavan, Sridhar Rajagopalan, Raymie Stata, Andrew Tomkins, and Janet L. Wiener. Graph structure in the web. *Computer Networks*, 33(1-6):309–320, 2000.
[8] T. Haveliwala and S. Kamvar. The second eigenvalue of the google matrix. Technical Report 2003-20, Stanford University, 2003. http://dbpubs.stanford.edu/pub/2003-20.
[9] Taher H. Haveliwala. Efficient computation of pagerank. Technical Report 1999-31, Stanford University, 1999. http://dbpubs.stanford.edu:8090/pub/1999-31.
[10] Taher H. Haveliwala. Topic-sensitive pagerank. In *Proc. of the 11th WWW Conference (WWW11)*, pages 517–526, 2002.
[11] Glen Jeh and Jennifer Widom. SimRank: A measure of structural-context similarity. In *Proc. 8th ACM SIGKDD Intl. Conf. on Knowledge Discovery and Data Mining*, July 2002.
[12] Sepandar D. Kamvar, Taher H. Haveliwala, Christopher D. Manning, and Gene H. Golub. Exploiting the block structure of the web for computing pagerank. Technical Report 2003-17, Stanford University, 2003. http://dbpubs.stanford.edu:8090/pub/2003-17.
[13] Robert Endre Tarjan. Depth-first search and linear graph algorithms. *SIAM J. Comput.*, 1(2):146–160, 1972.
[14] Yuan Wang and David J. DeWitt. Computing pagerank in a distributed internet search engine system. In *VLDB*, pages 420–431, 2004.
[15] Webbase project. Stanford University, http://www-diglib.stanford.edu/~testbed/doc2/WebBase/.
[16] Webgraph. University of Milano, http://webgraph.dsi.unimi.it/.