Internet Mathematics Vol. 2, No. 3: 387-404

Paradoxical Effects in PageRank Incremental Computations

Paolo Boldi, Massimo Santini, and Sebastiano Vigna

Abstract. Deciding which kind of visiting strategy accumulates high-quality pages more quickly is one of the most often debated issues in the design of web crawlers.

This paper proposes a related, and previously overlooked, measure of effectiveness for crawl strategies: whether the graph obtained after a partial visit is in some sense representative of the underlying web graph as far as the computation of PageRank is concerned. More precisely, we are interested in determining how rapidly the computation of PageRank over the visited subgraph yields node orders that agree with the ones computed in the complete graph; orders are compared using Kendall's τ.

We describe a number of large-scale experiments that show the following paradoxical effect: visits that gather PageRank more quickly (e.g., highest-quality first) are also those that tend to miscalculate PageRank. Finally, we perform the same kind of experimental analysis on some synthetic random graphs, generated using well-known web-graph models: the results are almost opposite to those obtained on real web graphs.

1. Introduction

The search for the best crawling strategy (i.e., a strategy that gathers early pages of high quality) is by now an almost classic research topic (see, e.g., [Cho et al. 98]). Being able to collect quickly high-quality pages is one of the major design goals of a crawler; this issue is particularly important, because, as noted

1542-7951/05 $0.50 per page

in [Eiron et al. 04], *even after crawling well over a billion pages, the number of uncrawled pages still far exceeds the number of crawled pages.*[1]

Usually, a basic measure of quality is PageRank [Page et al. 98], in one of its many variants. Hence, as a first step we can compare two strategies by looking at how fast the cumulative PageRank (i.e., the sum of the PageRank values of all the pages visited up to a certain point) grows over time. Of course, this comparison can only be performed after the end of the crawl, because one needs the whole graph to compute PageRank.

The most classical visit strategies are the following:

- *Depth-first* order: the crawler chooses the next page as the *last* that was added to the frontier; in other words, the visit proceeds in a LIFO fashion.
- *Random* order: the crawler chooses randomly the next page from the frontier.
- *Breadth-first* order: the crawler chooses the next page as the *first* that was added to the frontier; in other words, the visit proceeds in a FIFO fashion.
- *Omniscient* order (or quality-first order): the crawler uses a queue prioritised by PageRank values [Cho et al. 98]; in other words, it chooses to visit the page with highest quality among the ones in the frontier. This visit is meaningless unless a previous PageRank computation of the entire graph has been performed *before* the visit, but it is useful for comparisons. A variant of this strategy may also be adopted if we have already performed a crawl and thus we have the (old) PageRank values of (at least some of the) pages.

Both common sense and experiments (see, in particular, [Boldi et al. 05]) suggest that the visits listed above accumulate PageRank in an increasingly quicker way. This is to be expected, as the omniscient visit will point immediately to pages of high quality. The fact that the breadth-first visit yields high-quality pages was noted in [Najork and Wiener 01].

There is, however, a different and also quite relevant problem that has been previously overlooked in the literature: if we assume that the crawler has no previous knowledge of the web region it has to crawl, it is natural that it will try to detect page quality during the crawl itself, by computing PageRank on the region it has just seen. We would like to know whether a crawler doing so will obtain reasonable results or not. This question is even more urgent than it

[1] In the present paper by "uncrawled page" we are referring only to those pages that actually exist and would be crawled sometime in the future if the crawl goes on forever.

might appear at first, because we will most probably stop the crawl at a certain point anyway and use the graph that we have crawled to compute PageRank.

To attack this problem, we must establish a measure of how good a PageRank approximation computed on a subgraph is. Comparing directly PageRank values (for example, using L_1 norms) would be of course meaningless because of normalisation. However, we can compare two orders[2] on the subgraph:

- the order induced by PageRank values computed on the subgraph itself;
- the order induced by PageRank values computed on the whole graph.

In this paper, we use Kendall's τ on PageRank values as a measure of concordance between the two orders (a similar approach was followed in [Kumar et al. 00] to motivate the usefulness of a hierarchical algorithm to compute PageRank). The results that we obtain are paradoxical: *many strategies that accumulate PageRank quickly explore subgraphs having badly correlated orders*, and vice versa. Even more interestingly, these behaviours have not been reproduced by random graphs generated with two well-known models.

2. Kendall's τ and Its Computation

There are many correlation indices that can be used to compare orders.[3] One of the most widely used and intuitive is Kendall's τ; this classical nonparametric correlation index has recently received much attention within the web community for its possible applications to rank aggregation [Fagin et al. 03b, Fagin et al. 03a, Dwork et al. 01] and determining the convergence speed in the computation of PageRank [Kamvar et al. 03b].

Kendall's τ is usually defined as follows[4]:

Definition 2.1. [Kendall 55] Let $r_i, s_i \in \mathbf{R}$ $(i = 1, 2, \ldots, n)$ be two lists of rank values. Given a pair of distinct indices $1 \leq i, j \leq n$, we say that the pair is

- *concordant* iff $r_i - r_j$ and $s_i - s_j$ are both nonzero and have the same sign;
- *discordant* iff $r_i - r_j$ and $s_i - s_j$ are both nonzero and have opposite signs;
- an *r-tie* iff $r_i - r_j = 0$;

[2] Technically, a list of values induces a total *preorder*, not necessarily a total order, due to the possible presence of ties.

[3] For a thorough account on this topic, consult, for example, [Kendall and Gibbons 90].

[4] There are actually various subtly different definitions of τ, depending on how ties should be treated. The definition that we use here is usually referred to as τ_b.

- an *s-tie* iff $s_i - s_j = 0$;
- a *joint tie* iff $r_i - r_j = s_i - s_j = 0$.

Let C, D, T_r, T_s, and J be the number of concordant pairs, discordant pairs, r-ties, s-ties, and joint ties, respectively; also let $N = n(n-1)/2$. Of course, $C + D + T_r + T_s - J = N$. Kendall's τ of the two lists is now defined by

$$\tau = \frac{C - D}{\sqrt{(N - T_r)(N - T_s)}}.$$

Kendall's τ is always in the range $[-1, 1]$: $\tau = 1$ happens iff there are no non-joint ties and the two total orders induced by the lists are the same; $\tau = -1$, conversely, happens iff there are no non-joint ties and the two total orders are opposite of each other; thus, $\tau = 0$ can be interpreted as lack of correlation.

2.1. Knight's Algorithm

Apparently, evaluating τ on large data samples is not so common, and there is a remarkable scarcity of literature on the subject of computing τ in an efficient way. Clearly, the brute-force $O(n^2)$ approach is easy to implement but inefficient. Knight [Knight 66] presented in the sixties an $O(n \log n)$ algorithm for the computation of τ, but the only implementation of which we are aware belongs to the SAS system. Ideally, sophisticated data structures such as those described by Dietz [Dietz 82] could give an $O(n \log n / \log \log n)$ bound [Andersson and Petersson 98], but this approach is unfeasible in practice when the data set is so large that the lists occupy a significant fraction of the available core memory.

These considerations have been our motivation to write a direct, efficient implementation of Knight's algorithm, with a special variant needed to treat ties as required by our definition. This variant was indeed briefly sketched at the end of Knight's original paper, but details were completely omitted.[5] Besides the storage required by the two lists, our implementation just needs a support array of the same size as the lists (being based on a merge sort).

First of all, we sort the indices $\{1, 2, \ldots, n\}$ using r_i as first key and s_i as secondary key. More precisely, we compute a permutation π of the indices such that $r_{\pi(i)} \leq r_{\pi(j)}$ whenever $i \leq j$ and, moreover, if $r_{\pi(i)} = r_{\pi(j)}$ but $s_{\pi(i)} < s_{\pi(j)}$, then $i < j$.

Once the elements have been sorted, a linear scan is sufficient to compute T_r: a maximal sequence of indices $i, i+1, \ldots, j$ such that $r_{\pi(i)} = r_{\pi(i+1)} = \cdots = r_{\pi(j)}$

[5]The complete Java code of our implementation of Knight's algorithm is available under the GNU General Public License.

determines exactly $\binom{j-i+1}{2}$ r-ties. Moreover, with another linear scan and in a completely analogous way, we can compute J (this time, we look for maximal intervals where *both* r and s are constant). After the computation of D (described below), the indices will be sorted using s_i, so we shall be able to compute T_s in a similar fashion.

The knowledge of D is now sufficient to deduce τ; the computation of D is indeed the most interesting part of Knight's algorithm. Remember that all indices are at this point sorted using r_i as first key and s_i as secondary key. We apply a stable merge sort to all indices using s_i as the key. Every time we move an item *forward* during a merge, we increase D by the number of skipped items.

For instance, sorting the sequence

$$7, 2, 0, 6, 4, 3, 5, 1$$

requires first sorting subsequences of length two:

$$2, 7, 0, 6, 3, 4, 1, 5,$$

which contributes three discordances. Then, we merge them into subsequences of length four:

$$0, 2, 6, 7, 1, 3, 4, 5.$$

In doing so, we moved 0 forward by two positions, 6 forward by one position, and 1 forward by two positions, which contributes five more discordances. In the last merge, the list becomes sorted, so 1 is moved forward by three positions, whereas 3, 4, and 5, are moved forward by two. All in all, we obtain 17 discordances.

Note that the same amount can be more easily calculated using a bubble sort (the idea of using bubble sort appears in Kendall's original paper [Kendall 38]), but you cannot expect to run bubble sort on a graph with 100 million nodes.

3. Measuring the Quality of a Page with PageRank

PageRank [Brin and Page 98] is one of the best-known methods for measuring page quality; it is a static method (in that it does not depend on a specific query but rather it measures the absolute authoritativeness of each page), and it is based purely on the structure of the links, or, if you prefer, on the web graph. As it was recently noticed [Eiron et al. 04], PageRank is actually a set of ranking methods that depends on some parameters and variants; its most common version can be described as the behaviour of a random surfer walking through the web and depends on a single parameter $\alpha \in (0, 1)$ (the *damping factor*).

The random surfer, at each step t, is in some node p_t of the graph. The node p_{t+1} where the surfer will be in the next step is chosen as follows:

- if p_t had no outgoing arcs, p_{t+1} is chosen uniformly at random among all nodes;
- otherwise, with probability α, one of the arcs going out of p_t is chosen (with uniform distribution), and p_{t+1} is the node where the arc ends; with probability $1-\alpha$, p_{t+1} is once more chosen uniformly among all nodes.

The PageRank of a given node x is simply the fraction of time that the random surfer spent in x and will be denoted by $\mathrm{PR}_G(x)$.

An equivalent, maybe more perspicuous, way of defining PageRank is the following. Let $A = (a_{ij})$ be an $n \times n$ matrix (n being the number of nodes in the graph) defined as follows:

- if i has no outgoing arcs, $a_{ij} = 1/n$;
- if i has $d > 0$ outgoing arcs, and (i, j) is an arc, then $a_{ij} = \alpha/d + (1-\alpha)/n$;
- if i has $d > 0$ outgoing arcs, but (i, j) is not an arc, then $a_{ij} = (1-\alpha)/n$.

The matrix A turns out to be aperiodic, irreducible, and stochastic, so there is exactly one probability row vector r satisfying

$$rA = r.$$

The PageRank of node x is then exactly r_x.

The computation of PageRank is a rather straightforward task, which can be accomplished with standard tools of linear algebra [Haveliwala et al. 99, Kamvar et al. 03b]. The damping factor α influences both the convergence speed and the results obtained, but it is by now customary to choose $\alpha \approx 0.85$.

4. Using Kendall's τ to Contrast Crawl Strategies

The value assigned by PageRank to a page is not important *per se*; it is the relative order of pages with respect to PageRank that is actually interesting for search engines. This observation will guide us in what follows.

Consider the following typical scenario: G, the "real" web graph (unknown to the crawler), has N nodes; we perform some crawl of G, which determines a certain node order (the visit order) $x_1, x_2, \ldots, x_N$. For each $n = 1, 2, \ldots, N$, let $G_n = G[\{ x_1, x_2, \ldots, x_n \}]$ be the subgraph of G induced by the first n visited nodes. In other words, G_n is the graph collected at the nth step of the traversal.

In a real-world crawler, we will most probably stop the crawl at a certain point, say, after n pages have been crawled, typically with $n \ll N$. If you run PageRank on G_n, you obtain values that usually will not coincide with the values that those pages have when PageRank is run on G. Even worse, their *relative order* will be different.

Let τ_n be defined as Kendall's τ computed on $\mathrm{PR}_{G_n}(x_1)$, $\mathrm{PR}_{G_n}(x_2)$, ..., $\mathrm{PR}_{G_n}(x_n)$ and $\mathrm{PR}_G(x_1)$, $\mathrm{PR}_G(x_2)$, ..., $\mathrm{PR}_G(x_n)$. Clearly, $\tau_N = 1$, as at the end of the crawl G is entirely known, but the sequence $\tau_1, \tau_2, \ldots, \tau_N$ (hereafter referred to as the τ *sequence of the visit*) only depends on the visit order; it is not necessarily monotonic, but its behaviour can be used as a measure of how good (or bad) the chosen strategy is with respect to the way in which the order induced by PageRank is approximated during the crawl. The main goal of this paper is to present experimental results comparing τ sequences produced by different visit strategies. We shall also consider τ sequences of subgraphs not obtained through visits and use them for comparison.

Observe that a τ sequence has nothing to do with the convergence speed of PageRank, but rather with its tolerance to graph modifications, or, if you prefer, with its stability. There is a rich stream of research (see, for example, [Lee and Borodin 03, Ng et al. 01, Abiteboul et al. 03, Bianchini et al. 05, Langville and Meyer 04, Lempel and Moran 04]) concerning the robustness of PageRank with respect to graph modifications (node and/or link perturbation, deletion and insertion). Many authors observed, in particular, that PageRank is quite stable under graph modifications [Bianchini et al. 05], at least when the changing pages do not have high PageRank; even removing a large portion of pages, as many experiments suggest [Ng et al. 01], turns out to have small influence on PageRank. Some papers also propose variants of the PageRank algorithm [Abiteboul et al. 03] that can be computed adaptively and robustly with respect to graph evolution.

However, most of these studies (with the exception of [Lempel and Moran 04]) measure the stability of PageRank by computing the distance (usually, under L^1 or L^2 norm) between the PageRank vectors. However interesting this metric might be, it is not directly related to our measure. One might object that τ is not a completely fair way to analyse PageRank computation: after all, two PageRank vectors may turn out to be poorly correlated only because of small value fluctuations in pages with low PageRank. In other words, since τ is not continuous, small perturbations in the PageRank vector may have disastrous effects.

Nevertheless, the real-world usage of PageRank is, by its very nature, discontinuous—PageRank is essentially used only to order pages, its real numerical value being irrelevant. Moreover, even if fluctuations may only concern low-

ranked pages, thus not affecting the mutual order of the top pages, this fact should not be underestimated; if a query is satisfied only by a set of pages of low rank, their relative order will be for the user as worthy as the order of the top pages. The latter effect is noticeable not only for some esoteric queries, but typically also for relevant queries of interest only to a small community: you will not expect that a query like *group torsion* or *analog editing* is satisfied by high-ranked pages, but the relative order of the pages that satisfy those queries is certainly meaningful.

5. Experimental Setup

5.1. Graphs

We based our experiments on four web graphs:

- a 41,291,594-node snapshot of the Italian domain `.it`, performed with UbiCrawler [Boldi et al. 05] (later on referred to as the *Italian graph*);
- a 118,142,155-node graph obtained from the 2001 crawl performed by the WebBase crawler [Hirai et al. 00] (later on referred to as the *WebBase graph*);
- a 10,000,000-node synthetic graph generated using the Copying Model proposed in [Kumar et al. 00], with suitable parameters and rewiring (later on referred to as the *copying-model graph*);
- a 10,000,000-node synthetic graph generated using the Evolving Network Model proposed in [Albert et al. 99], with suitable parameters and rewiring (later on referred to as the *evolving-model graph*).

The two synthetic graphs are much smaller than the others, because by their random nature there would be no gain in using a larger node set.

For each graph, we computed the strongly-connected components and discarded the nodes that were not reachable from the giant component: we decided to do so because otherwise a single visit would not collect the whole graph, and the visit order would not depend only on the visit strategy and on a single starting node. The resulting graph sizes were 41,291,594 for the Italian graph, 95,688,816 for the WebBase graph, 4,863,948 for the copying-model graph, and 5,159,894 for the evolving-model graph.

5.2. Visit Seeds

For each of the four graphs in Section 5.1, we performed visits starting from three different seeds (a.k.a. starting URLs). We decided to perform visits from the highest-ranked, lowest-ranked, and median-ranked nodes within the giant component (choosing seeds within the giant component guarantees that the whole graph will be collected at the end of the visit).

5.3. Visit Strategies

For each of the graphs and every visit seed, we performed five different kinds of visits: `BF` (breadth-first traversal), `DF` (depth-first traversal), `RF` (random-first traversal: the next node to be visited is chosen at random from the frontier), `QF` (quality-first traversal: the next node to be visited is the one having the largest PageRank value in the frontier; here, by PageRank we mean PageRank in the real graph, not in the subgraph), and `IQF` (inverse-quality-first traversal: in this case, we choose instead the node having the *smallest* PageRank value in the frontier).

5.4. Sampling

Computing PageRank and Kendall's τ on a large graph is a heavy computation task. Due to the large size of our data and to the large number of different visits, we decided to spread logarithmically our sample points, so as to avoid sampling too many times very large graphs. In general, samples are multiplicatively spaced at least by $\sqrt[64]{2}$, albeit in several cases we used a thicker sampling. Note also that when the subgraph sizes exceed half of the original graph size, all curves become smooth, so using denser samples in that region is probably useless.

5.5. Computational Time

We performed our PageRank and τ computation on five dual-processor PCs and two multi-processor servers (the latter were essential in handling the largest samples). Overall, we used about 1600 hours of CPU user time.

5.6. Tools

All our tests were performed using the WebGraph framework [Boldi and Vigna 04, Boldi and Vigna 05]. Moreover, at the address http://webgraph-data.dsi.unimi.it/, you may find both the Italian and the WebBase graphs (recall, however, that we only used the portion of WebBase that can be reached from the giant component). The two synthetic graphs were produced using COSIN [Donato et al. 04].

6. Experimental Results

6.1. Comparing τ Sequences

Figure 1 represents the τ sequences obtained during the visits of the four graphs, starting from the node with largest PageRank value within the giant component

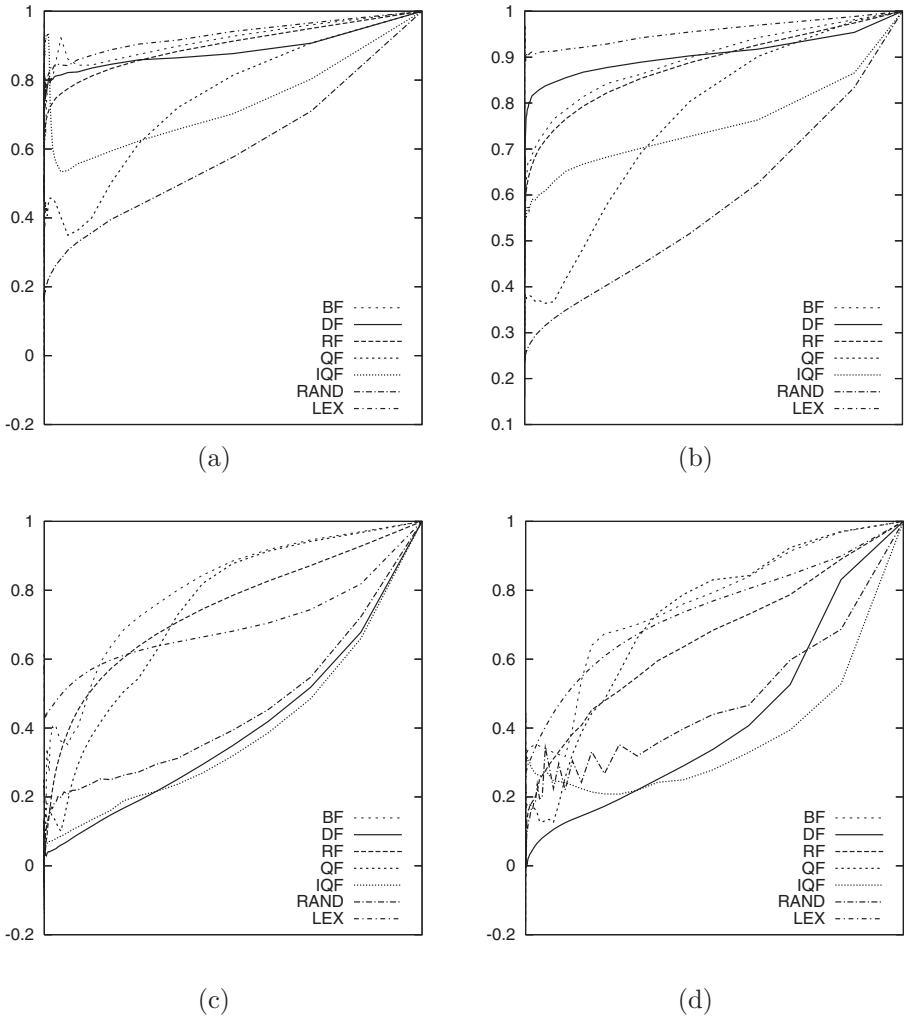

Figure 1. τ sequences for (a) the Italian graph, (b) the WebBase graph, (c) the copying-model graph, and (d) the evolving-model graph, starting from the highest-ranked node in the giant component.

(the graphs for the other seeds do not differ significantly). In each graph, we also plotted the τ sequence `RAND` corresponding to a random node order (note that this is quite different from a random-first traversal, because it is not a graph visit); additionally, the diagrams contain an extra line, called `LEX`, whose meaning will be explained in the next section. The horizontal axis represents time (or, more precisely, the number of nodes that have been visited), whereas the vertical axis represents the value of τ.

6.2. Cumulative PageRank

For comparison, we computed the cumulative PageRank (the sum of all PageRanks of the pages collected so far) during the visits of the Italian graph (Figure 2).

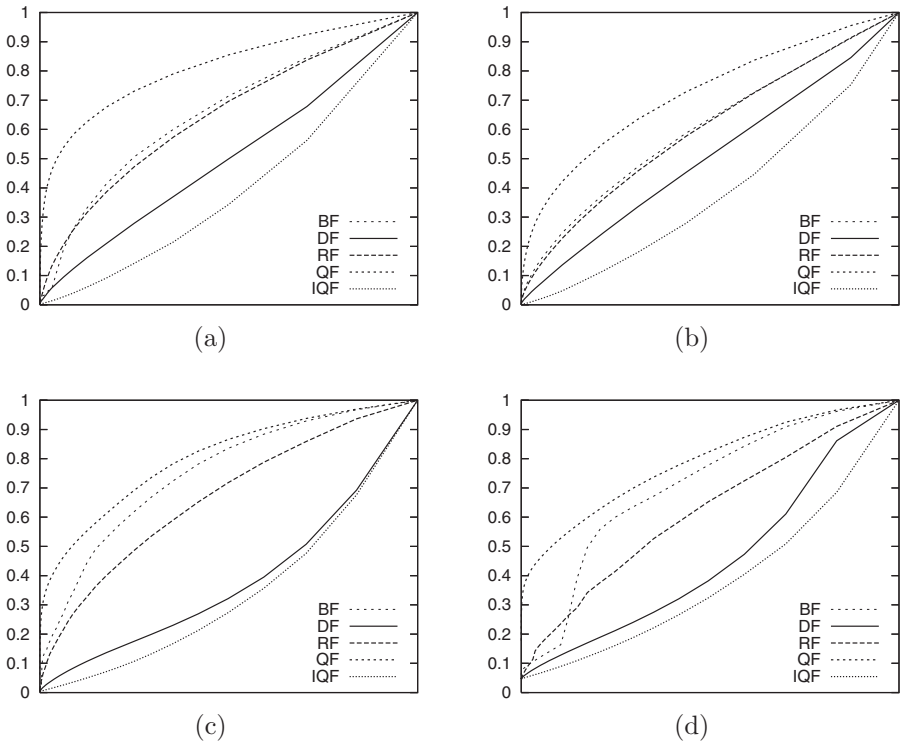

Figure 2. Cumulative PageRank obtained during a visit of (a) the Italian graph, (b) the WebBase graph, (c) the copying-model graph, and (d) the evolving-model graph, starting from the highest-ranked node in the giant component.

7. Interpretation of the Experimental Results

7.1. Cumulative PageRank

The graph of cumulative PageRank (Figure 2) confirms folklore and experiments reported by the literature: if you want to collect pages with high PageRank value rapidly, you should use a breadth-first (BF) traversal; this strategy is overcome only by the omniscient quality-first (QF) traversal. Obviously, using an inverse-quality-first (IQF) traversal produces the poorest results. Also, depth-first (DF) is only marginally better than IQF, whereas, a bit surprisingly, RF is very good, even though worse than BF. As in the case of τ sequences, the results are not significantly influenced by the seed.

7.2. PageRank: Locality and Collective Nature

We now turn to the study of the τ sequences (Figure 1). Looking at the diagrams of the Italian and of the WebBase graphs, we observe the following facts.

1. Even though τ sequences do not increase monotonically, they anyway tend to grow quite steadily, after a brief chaotic transient.

2. DF and BF are, in most cases, comparable; often, DF is even better. Moreover, they are both quite close to RF. Note that these behaviours are in sharp contrast with the graphs for cumulative PageRank, where BF is far better than DF.

3. QF, which used to be the best strategy for cumulative PageRank, is now the worst possible strategy in the first part of the crawl—even IQF is better!

4. All the visit strategies considered are anyway more τ-effective than collecting nodes at random: the line of RAND is constantly below all other lines, in all graphs.

Apparently, the roles of visit strategies are now completely scrambled. Note that the situation is rather robust: it is the same for all seeds and for both graphs, and we indeed tested it against other, smaller real-world examples obtaining quite similar behaviours.

To try to understand the very reasons behind the observed facts, it might be helpful to look at the diagrams in logarithmic (horizontal) scale (see Figure 3). Under this scale, one observes that all visits present an initial burst that is absent in RAND, whose growth is quite steady. Our explanation for this fact is that, at the beginning of a visit, the crawl tends to remain confined within a single web site (or a small set of web sites).

Indeed, a recent work [Kamvar et al. 03a] makes an analysis quite similar to ours with a different purpose: they prove that PageRank computation can be made more efficient by computing local PageRanks first, where "local" means that the computation is performed on (the subgraph induced by) the nodes of a single web site. The authors motivate their algorithm by showing that lo-

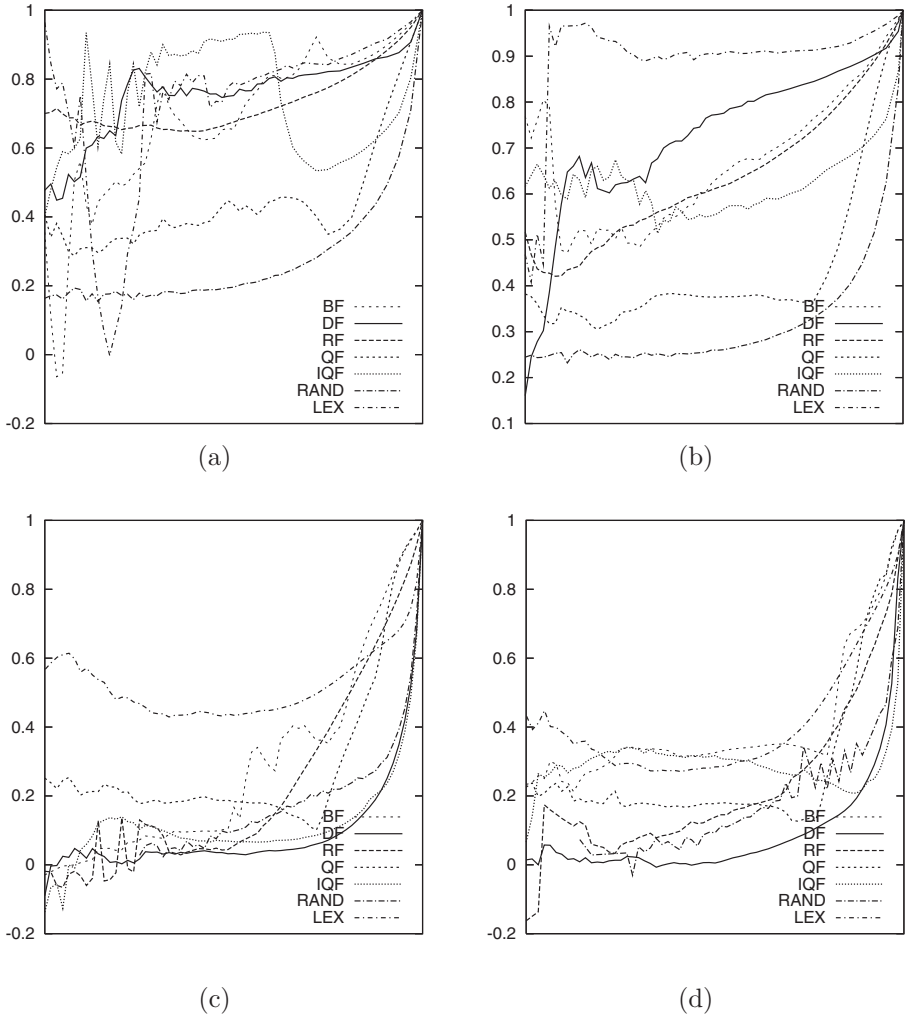

Figure 3. τ sequences in horizontal logarithmic scale for (a) the Italian graph, (b) the WebBase graph, (c) the copying-model graph, and (d) the evolving-model graph, starting from the highest-ranked node in the giant component.

cal PageRanks agree when they are computed on the local subgraph instead of the whole graph, and they use (a version of) Kendall's τ to measure agreement. (They perform their experiments on a small 683,500-page snapshot of the Stanford/Berkeley site.)

Essentially, inside a single site, PageRank depends heavily on the local links. This phenomenon, that we might call *PageRank locality*, explains why any visit performs better than a random selection of nodes.

With the aim of obtaining an empirical confirmation for our conjecture of PageRank locality, we tried another experiment. We computed the τ sequence of subsets of nodes collected in lexicographic order; in other words, we started with the first URL (in lexicographic order) and proceeded collecting URLs in this way. The resulting τ sequences are shown in the diagram as `LEX`: since `LEX` is the most local way of collecting nodes, we expect the τ sequence to be distinctly good. And, indeed, `LEX` lies always above all other lines.

Locality can also explain the success of `DF` (which beats `BF` in the first half of the WebBase graph). Depth-first visits tend to wander in a site, and then, as soon as a cross-site link is chosen, jump to another site, wander there for a while, and so on. This behaviour is closer to `LEX` than to a breadth-first visit.

However, locality cannot explain other phenomena; for example, why can an inverse quality visit beat for a long time an omniscient visit? We believe that the paradoxical behaviour of priority-based visits can only be explained by the *collective nature* of PageRank. To have a good τ-correlation with PageRank, besides being local, a subgraph must also be representative of the collection of pages, that is, *sufficiently random* (with respect to PageRank ordering). In other words, *high authoritativeness cannot be achieved without the help of minor pages.* Selecting blindly the most authoritative pages leads to a world where authority is subverted. Again, even the opposite choice—selecting the pages with lowest authority—works better sometimes.[6]

7.3. Random Models

The graphs for the two chosen random models are quite disappointing. The PageRank-cumulation behaviour (Figure 2) follows in a relatively close way what happens for real graphs (albeit the models are somehow too good to be true—breadth-first visits beat by far and large random visits, which is not true on real graphs). When, however, we turn to τ-sequences (Figure 1), we find marked differences. For example, `DF` works much more poorly than it does in real web graphs, whereas `QF` seems very powerful while it is not (note that `LEX` is now

[6]This observation should lead us to reconsider gossiping versus good newspapers as a means for selecting authoritative sources of information.

almost meaningless, as nodes have no particular order: the fact that LEX is better than RAND is a pure artifact of the models). A partial explanation for this is that the chosen models do not take locality into account, hence failing to show the behaviours discussed above. Certainly there is a lot of room for improvement.

8. Conclusions and Further Work

Our results, as the title suggests, are somewhat paradoxical: strategies that work quite well when you want to accumulate pages with high quality in a short time tend to behave rather poorly when you try to approximate PageRank on a partial crawl, and vice versa. In particular, pure breadth-first is not a good strategy (for completely different reasons, the authors of [Castillo et al. 04] have been led to a similar conclusion). Our explanation for this paradox is tentative: the goal of this paper is to raise questions more than to give answers.

To conclude, we address some issues about our methodology and some possible further directions.

- *Treatment of dangling links.* We computed the PageRank of a subgraph ignoring the fact that some nodes of the subgraph would contain many outlinks that are not there simply because our crawl is partial. Some authors recently suggested [Eiron et al. 04] to modify the definition of PageRank to take this phenomenon into account. It would be interesting to know whether their version of PageRank makes things better or not.

- *Comparing only top nodes.* Another possible direction is considering a different metric that does not take into account the whole order, but only the relative order of the top k elements, where k is either constant or a fixed fraction of nodes; this approach might also be modified so as to use a definition of τ that works also when the orders are not defined over the same set of nodes, like the ones given in [Fagin et al. 03b]. Note, however, that in real-world applications the whole order is very important (even more important than the order of high-quality nodes, as explained at the end of Section 4).

- *More strategies.* Our results would suggest the implementation of crawl strategies that work effectively with respect to both PageRank and τ. A natural candidate strategy would be a mixed order, using a depth-first intra-site visit and operating breadth-first for extra-site links. This strategy would be quite natural for a parallel or distributed crawler, in that every worker might proceed in depth-first order within a site (respecting, of course, the usual politeness assumptions) whereas out-of-site links

might be treated in a breadth-first manner: UbiCrawler [Boldi et al. 05] was originally designed to work with this strategy. Another interesting strategy could be performing a crawl in lexicographic order (the next node to be visited is the first node of the frontier in lexicographic order).

- *More random models.* The two graph models that we have tried presented behaviors that did not quite match with real-world graphs. A natural investigation would be trying to discover if other, more sophisticated models work better.

Acknowledgements. This work has been partially supported by MIUR COFIN "Linguaggi formali e automi: metodi, modelli e applicazioni" and by a "Finanziamento per grandi e mega attrezzature scientifiche" of the Università degli Studi di Milano.

References

[Abiteboul et al. 03] Serge Abiteboul, Mihai Preda, and Gregory Cobena. "Adaptive On-Line Page Importance Computation." In *Proceedings of the 12th International Conference on World Wide Web*, pp. 280–290. New York: ACM Press, 2003.

[Albert et al. 99] Reka Albert, Albert-Laszlo Barábasi, and Hawoong Jeong. "Diameter of the World Wide Web." *Nature* 401 (September 1999), 130–131.

[Andersson and Petersson 98] Arne Andersson and Ola Petersson. "Approximate Indexed Lists." *J. Algorithms* 29:2 (1998), 256–276.

[Bianchini et al. 05] Monica Bianchini, Marco Gori, and Franco Scarselli. "Inside PageRank." *ACM Transactions on Internet Technologies* 5:1 (2005), 92–128.

[Boldi and Vigna 04] Paolo Boldi and Sebastiano Vigna. "The WebGraph Framework I: Compression Techniques." In *Proceedings of the 13th International Conference on World Wide Web*, pp. 595–601. New York: ACM Press, 2004.

[Boldi and Vigna 05] Paolo Boldi and Sebastiano Vigna. "Codes for the World Wide Web." *Internet Math.* 2:4 (2005), 405–427.

[Boldi et al. 05] Paolo Boldi, Bruno Codenotti, Massimo Santini, and Sebastiano Vigna. "Ubicrawler: A Scalable Fully Distributed Web Crawler." *Software: Practice & Experience* 34:8 (2004), 711–726.

[Brin and Page 98] Sergey Brin and Lawrence Page. "The Anatomy of a Large-Scale Hypertextual Web Search Engine." *Computer Networks* 30:1–7 (1998), 107–117.

[Castillo et al. 04] Carlos Castillo, Mauricio Marin, Andrea Rodriguez, and Ricardo Baeza-Yates. "Scheduling Algorithms for Web Crawling." In *WebMedia and LA-WEB 2004*, pp. 10–17. Los Alamitos, CA: IEEE Press, 2004.

[Cho et al. 98] Junghoo Cho, Hector García-Molina, and Lawrence Page. "Efficient Crawling Through URL Ordering." *Computer Networks and ISDN Systems* 30:1–7 (1998), 161–172.

[Dietz 82] Paul F. Dietz. "Maintaining Order in a Linked List." In *Proceedings of the Fourteenth Annual ACM Symposium on Theory of Computing*, pp. 122–127. New York: ACM Press, 1982.

[Donato et al. 04] Debora Donato, Luigi Laura, Stefano Leonardi, and Stefano Milozzi. "A Library of Software Tools for Performing Measures on Large Networks." Available from World Wide Web (http://www.dis.uniroma1.it/~cosin/html_pages/COSIN-Tools.htm), 2004.

[Dwork et al. 01] Cynthia Dwork, Ravi Kumar, Moni Naor, and D. Sivakumar. "Rank Aggregation Methods for the Web." In *Proceedings of the 10th International Conference on World Wide Web*, pp. 613–622. New York: ACM Press, 2001.

[Eiron et al. 04] Nadav Eiron, Kevin S. McCurley, and John A. Tomlin. "Ranking the Web Frontier." In *Proceedings of the 13th International Conference on World Wide Web*, pp. 309–318. New York: ACM Press, 2004.

[Fagin et al. 03a] Ronald Fagin, Ravi Kumar, Kevin S. McCurley, Jasmine Novak, D. Sivakumar, John A. Tomlin, and David P. Williamson. "Searching the Workplace Web." In *Proceedings of the 12th International Conference on World Wide Web*, pp. 366–375. New York: ACM Press, 2003.

[Fagin et al. 03b] Ronald Fagin, Ravi Kumar, and D. Sivakumar. "Comparing Top k Lists." In *Proceedings of the 14th Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 28–36. Philadelphia, PA: Society for Industrial and Applied Mathematics, 2003.

[Haveliwala et al. 99] T. Haveliwala. "Efficient Computation of PageRank." Technical report, Stanford University, 1999.

[Hirai et al. 00] Jun Hirai, Sriram Raghavan, Hector Garcia-Molina, and Andreas Paepcke. "WebBase: A Repository of Web Pages." *Computer Networks* 33:1–6 (2000), 277–293.

[Kamvar et al. 03a] 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, Stanford University, 2003.

[Kamvar et al. 03b] Sepandar D. Kamvar, Taher H. Haveliwala, Christopher D. Manning, and Gene H. Golub. "Extrapolation Methods for Accelerating PageRank Computations." In *Proceedings of the 12th International Conference on World Wide Web*, pp. 261–270. New York: ACM Press, 2003.

[Kendall 38] Maurice G. Kendall. "A New Measure of Rank Correlation." *Biometrika* 30:1–2 (1938), 81–93.

[Kendall 55] Maurice G. Kendall. *Rank Correlation Methods.* New York: Hafner Publishing Co., 1955.

[Kendall and Gibbons 90] Maurice Kendall and Jean Dickinson Gibbons. *Rank Correlation Methods.* London: Edward Arnold, 1990.

[Knight 66] William R. Knight. "A Computer Method for Calculating Kendall's Tau with Ungrouped Data." *Journal of the American Statistical Association* 61:314 (1966), 436–439.

[Kumar et al. 00] R. Kumar, P. Raghavan, S. Rajagopalan, D. Sivakumar, A. Tomkins, and E. Upfal. "Stochastic Models for the Web Graph." In *Proceedings of the 41st Annual Symposium on Foundations of Computer Science*, edited by Danielle C. Young, pp. 57–65. Los Alamitos, CA: IEEE Computer Society, 2000.

[Langville and Meyer 04] Amy N. Langville and Carl D. Meyer. "Deeper Inside PageRank." *Internet Mathematics* 1:3 (2004), 335–380.

[Lee and Borodin 03] Hyun Chul Lee and Allan Borodin. "Perturbation of the Hyper-Linked Environment." In *Computing and Combinatorics, 9th Annual International Conference, COCOON 2003, Big Sky, MT, USA, July 25–28, 2003, Proceedings*, pp. 272–283, Lecture Notes in Computer Science 2697. Berlin: Springer, 2003.

[Lempel and Moran 04] R. Lempel and S. Moran. "Rank-Stability and Rank-Similarity of Link-Based Web Ranking Algorithms in Authority Connected Graphs." *Information Retrieval* 8:2 (2005), 245–264.

[Najork and Wiener 01] Marc Najork and Janet L. Wiener. "Breadth-First Search Crawling Yields High-Quality Pages." In *Proceedings of 10th International Conference of World Wide Web*, pp. 114–118. New York: ACM Press, 2001.

[Ng et al. 01] Andrew Y. Ng, Alice X. Zheng, and Michael I. Jordan. "Stable Algorithms for Link Analysis." In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 258–266. New York: ACM Press, 2001.

[Page et al. 98] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. "The PageRank Citation Ranking: Bringing Order to the Web." Technical report, Stanford Digital Library Technologies Project, Stanford University, 1998.

Paolo Boldi, Dipartimento di Scienze dell'Informazione, Università degli Studi di Milano, via Comelico 39/41, I-20135 Milano, Italy (boldi@dsi.unimi.it)

Massimo Santini, Dipartimento di Scienze dell'Informazione, Università degli Studi di Milano, via Comelico 39/41, I-20135 Milano, Italy (santini@dsi.unimi.it)

Sebastiano Vigna, Dipartimento di Scienze dell'Informazione, Università degli Studi di Milano, via Comelico 39/41, I-20135 Milano, Italy (vigna@acm.org)

Received November 1, 2004; accepted August 31, 2005.