Journal of Discrete Algorithms 8 (2010) 199–213

Contents lists available at ScienceDirect

Journal of Discrete Algorithms

www.elsevier.com/locate/jda

Choose the damping, choose the ranking?

Marco Bressan *, Enoch Peserico

Department of Information Engineering, University of Padova via Gradenigo 6, 35131 Padova, Italy

ARTICLE INFO

Article history:
Received 2 December 2008
Accepted 1 November 2009
Available online 5 November 2009

Keywords:
Graph
Ranking
Link analysis
PageRank
Damping factor
Web
CiteSeer

ABSTRACT

To what extent can changes in PageRank's damping factor affect node ranking? We prove that, at least on some graphs, the top k nodes assume *all* possible $k!$ orderings as the damping factor varies, even if it varies within an *arbitrarily small* interval (e.g. $[0.84999, 0.85001]$). Thus, the rank of a node for a given (finite set of discrete) damping factor(s) provides very little information about the rank of that node as the damping factor varies over a continuous interval.

We bypass this problem introducing *lineage analysis* and proving that there is a simple condition, with a "natural" interpretation independent of PageRank, that allows one to verify "in one shot" if a node outperforms another *simultaneously* for all damping factors and all damping variables (informally, time variant damping factors). The novel notions of *strong rank* and *weak rank* of a node provide a measure of the fuzziness of the rank of that node, of the objective orderability of a graph's nodes, and of the quality of results returned by different ranking algorithms based on the random surfer model.

We deploy our analytical tools on a $41M$ node snapshot of the .it Web domain and on a $0.7M$ node snapshot of the CiteSeer citation graph. Among other findings, we show that rank is indeed relatively stable in both graphs; that "classic" PageRank ($d = 0.85$) marginally outperforms Weighted In-degree ($d \to 0$), mainly due to its ability to ferret out "niche" items; and that, for *both* the Web and CiteSeer, the ideal damping factor appears to be 0.8–0.9 to obtain those items of high importance to at least one (model of randomly surfing) user, but only 0.5–0.6 to obtain those items important to every (model of randomly surfing) user.

1. Introduction

This paper addresses the fundamental question of how the ranking induced by PageRank can be affected by variations of the damping factor. This introduction briefly reviews the PageRank algorithm (Section 1.1) and the crucial difference between score and rank (Section 1.2) before presenting an overview of our results and the organization of the rest of the paper (Section 1.3).

1.1. PageRank and the damping factor

PageRank [23] is probably the best known of *link analysis* algorithms, ranking nodes of a generic graph in order of "importance". PageRank was first used in search engines to rank the nodes of the Web graph; since then, it has been adapted to many other application domains, such as credit and reputation systems [17], crawling [8], automatic construction of Web

* Corresponding author.
E-mail addresses: bressanm@dei.unipd.it (M. Bressan), enoch@dei.unipd.it (E. Peserico).

1570-8667/$ – see front matter
doi:10.1016/j.jda.2009.11.001

directories [7], word stemming [2], automatic synonym extraction in a dictionary [4], word sense disambiguation [26], item selection [28] and text summarization [10].

In its original form, PageRank is based on a model of a Web surfer that probabilistically browses the Web graph, starting at a node chosen at random according to the distribution given by a *personalization vector* $\mathbf{e} > 0$ whose generic component e_v is the probability of starting at node v. At each step, if the current node has outgoing links to $m > 0$ other nodes $v_1, \ldots, v_m$, the surfer next browses with probability d one of those nodes (chosen uniformly at random), and with probability $1 - d$ a node chosen at random according to $\mathbf{e}$. If the current node is a sink with no outgoing links, the surfer automatically chooses the next node at random according to $\mathbf{e}$.

If the *damping factor* d is less than 1 (for the Web graph a popular value is ≈ 0.85 [20,23]), the probability $P_v(t)$ of visiting the node v at time t converges for $t \to \infty$ to a stationary probability P_v. This is the score PageRank assigns to v, ranking all nodes in order of non-increasing score. Intuitively, the more ancestors (both immediate and far removed) a node has, and the fewer descendants those ancestors have, the higher the score of that node.

It is crucial to observe that both the score of each node and its rank may in general change as the damping factor changes [25]. Lower damping factors decrease the likelihood of following long paths of links without taking a random jump — and thus increase the contribution to a node's score and rank of its more immediate ancestors.

1.2. Score vs. rank

PageRank assigns to the n nodes of a graph G a score vector that is the dominant (right) eigenvector of the stochastic $n \times n$ matrix $d\mathbf{M} + (1 - d)\mathbf{e} \cdot \mathbf{1}^T$, where $\mathbf{M}$ is obtained from the transposed adjacency matrix of G by normalizing each non-zero column, and replacing each zero column (corresponding to a sink) with the personalization vector $\mathbf{e}$. Typically, the score vector is calculated using the Power Method [20]. Thus, one can analyse the impact of the damping factor on the *score* vector using the highly developed toolset of linear algebra [20]. This is no longer true when dealing with *rank*: loss of linearity and continuity in the mathematical model make analysis considerably more difficult and require different tools.

And unfortunately, in many cases, (variation in) rank is far more important than (variation in) score. One might observe that the PageRank score is often combined with other scores (e.g. obtained from textual analysis in the context of Web pages) to provide a ranking, and thus large variations in PageRank score would seem more important than large variations in rank. However, this is (ever more) often not the case. Some search engines may simply filter items ranked by PageRank through a Boolean model of relevance to the query at hand — in these cases the ordering induced by PageRank is strictly maintained [2]. And many other applications simply use the unmodified PageRank score to rank items in order of precedence, whether for efficiency or by lack of other valid alternatives: e.g. Web crawlers [8] and reputation systems [17]. For a more thorough discussion of why, in many cases, rank is more important than score and thus why it is crucial to analyse the perturbation in rank induced by variations of the damping factor, see [13,16,21,22,24].

1.3. Our results

This paper addresses the fundamental question of how variations in PageRank's damping factor can affect the ranking of nodes: clearly this ranking elicits confidence only when relatively insensitive to small variations in the value of a parameter in the user model (the damping factor) likely to differ between different users and, in any case, hard to assess precisely. The rest of the paper is organized as follows.

Section 2 shows that for any k, at least on some graphs, *arbitrarily small* variations in the damping factor can completely reverse the ranking of the top k nodes, or indeed make them assume all possible $k!$ orderings. It is natural to ask whether this can happen in "real" graphs encountered in the vast and growing number of application domains of PageRank. Experiments sampling ranking for a handful of different damping factors [12,22] suggest this is not the case at least for the Web graph (leaving open the issue of other application domains). However, verifying rank stability for discrete variations in the damping factor (e.g. 0.01 increments) is not sufficient to conclude that rank is stable as the damping factor varies over a *whole continuous interval* — just like sampling the function $f(x) = \sin(100\pi x)$ for $x = 0.01, 0.02, \ldots$ is not enough to conclude that $f(x) = 0\ \forall x \in \mathbb{R}$.

Section 3 provides the analytical tools to address this issue. We show a simple, "natural" condition that is both necessary and sufficient to guarantee that a node outranks another *simultaneously* for all damping factors and all damping variables (informally, time-variant damping factors). This condition, based on the concept of *lineage* of a node, can be checked efficiently and has an intuitive justification totally independent of PageRank.

Section 4 leverages lineage analysis to introduce the novel concepts of *strong rank* and *weak rank*. These provide for each node a measure of the "fuzziness" of its ranking that is subtly but profoundly different from pure rank variation; they also provide a measure of the effectiveness of the random surfer model on a generic graph; finally, they allow an objective evaluation of the performance of "classic" ($d = 0.85$) PageRank and some of its variations (e.g. $d \to 0$).

Section 5 brings the analytical machinery of Section 3 to bear on two real graphs — a snapshot of the .it domain, and the CiteSeer citation graph [9].

Section 6 summarizes our results, analyses their significance, and reviews a few problems this paper leaves open, before concluding with the bibliography.

2. The damping makes the ranking

This section presents two theorems showing that, at least on some graphs, a minuscule variation of the damping factor can dramatically change the ranking of the top k nodes. More formally, we prove:

Theorem 1. *For every even $k > 1$ and every d satisfying $\frac{1}{k} < d < 1 - \frac{1}{k}$, there is a graph G of $2k^2 + 4k - 2$ vertices such that PageRank's top k nodes are, in order, $\langle v_1, \ldots, v_k \rangle$ if the damping factor is d, and $\langle v_k, \ldots, v_1 \rangle$ if it is $d + \frac{1}{k}$.*

Theorem 2. *Consider an arbitrary set Π of orderings of k nodes $v_1, \ldots, v_k$ ($|\Pi| \leqslant k!$), and an arbitrary open interval $\mathcal{I} \subset [0, 1]$, however small. Then there is a graph G such that PageRank's top k nodes are always $v_1, \ldots, v_k$ but appear in every order in Π as the damping factor varies within $\mathcal{I}$.*

By Theorem 1, there are graphs of $\approx 2M$ nodes (a size comparable to that of a citation archive or of a small first level domain) where a variation of the damping factor as small as 0.001 (e.g. from 0.850 to 0.851) can cause a complete reversal of the ranking of the top 1000 items; and the sensitivity to the damping factor can grow even higher for larger graphs. Theorem 2 is even more general: for any arbitrarily small interval of variation of the damping factor, there are graphs in which the top items assume *all* possible permutations (but providing bounds on the size of these graphs is beyond the scope of this paper).

The rest of this section is organized as follows. Section 2.1 introduces some notation, necessary for the proofs of these two theorems, that will also be of use later. Section 2.2 is devoted to the proofs themselves, and may be skipped without impairing the understanding of the rest of the paper.

2.1. Notation

The score assigned by PageRank to the node v using a damping factor d, $P_v(d)$ — i.e. the stationary probability of being on that node according to the model presented in Section 1.1 — can be seen as the sum, over all $\ell \geqslant 0$, of the probability of taking, ℓ timesteps in the past, the last random jump to a node v_ℓ at distance ℓ from v, and following any path of length ℓ from v_ℓ to v.

More formally, denote by $p \xrightarrow{\ell} v$ the fact that p is a path of $\ell + 1$ vertices $\langle v_\ell, \ldots, v_1, v \rangle$, from some vertex v_ℓ to v. Let branching(p) be the inverse of the product of the out-degrees $\omega_\ell, \ldots, \omega_1$ of $v_\ell, \ldots, v_1$, i.e. branching(p) $= 1/(\omega_\ell \ldots \omega_1)$ (informally, the probability of following the whole path if one starts on the first node v_ℓ and no random jumps are taken). Then (see e.g. [3]):

$$P_v(d) = \sum_{\ell=0}^{+\infty} (1-d) d^\ell \sum_{p \xrightarrow{\ell} v} \text{branching}(p) \cdot \mathbf{e}_{v_\ell} \tag{1}$$

where $\mathbf{e}_{v_\ell}$ is the entry of $\mathbf{e}$ associated with the first node, v_ℓ, of the path p. It is easy to verify that the term $\sum_{p \xrightarrow{\ell} v} \text{branching}(p) \cdot \mathbf{e}_{v_\ell}$, the *branching contribution at level ℓ*, equals the sum of the row of $\mathbf{M}^\ell$ corresponding to v, each weighted by the corresponding component of $\mathbf{e}$. Intuitively, this term corresponds to the score a node v would hold after ℓ timesteps if each node v_i started with a score $\mathbf{e}_{v_i}$ and, at each timestep, bequeathed all of its score dividing it evenly among its children. Note that branching contribution is completely independent of the damping factor; whereas the other term of the product, $(1-d)d^\ell$ (the probability of having taken the last jump exactly ℓ timesteps in the past), depends solely on the damping factor and is independent of the structure of the graph.

2.2. Rank can change completely: A proof

Proof of Theorem 1. For simplicity we set the personalization vector $\mathbf{e} = [\frac{1}{n} \ldots \frac{1}{n}]$, but the result can be easily extended to the general case $\mathbf{e} > 0$. Let m be the smallest integer such that $d < \frac{k}{m} < d + \frac{1}{k}$ — one such $m \leqslant k^2$ always exists, since $\frac{k}{m} - \frac{k}{m+1} = \frac{k}{m(m+1)} < \frac{1}{k}$ for $\frac{k}{m} \leqslant 1 - \frac{1}{k}$. G is formed by 4 sets of nodes (see Fig. 1): $V = \{v_1, \ldots, v_k\}$ is the set of k nodes whose ranking is reversed by the variation in d; $W = \{w_1, \ldots, w_k\}$ and $U = \{u_1, \ldots, u_{k-1}, u'_1, \ldots, u'_{k-1}\}$ are two sets of nodes that are parents of nodes in V; $T = \{t_1, \ldots, t_{2k^2}\}$ is a set of nodes that are parents both of nodes in V and of nodes in W. Nodes of V are sinks; nodes of U and W have outdegree $\frac{k}{2}$ except for w_k that is a sink; nodes of T have outdegree k. Nodes of T and U have in-degree 0; nodes of W are all linked (only) by $t_1, \ldots, t_m$ (and thus all have the same score); nodes of V are all linked by $t_{m+1}, \ldots, t_{2k^2}$, and by one or more nodes of W and/or U (and thus always have a higher score than every other node). More specifically, v_i receives a link from each of $u_1, \ldots, u_{k-i}$ and $u'_1, \ldots, u'_{k-i}$ for $i \leqslant k/2$ and a link from each of $u_i, \ldots, u_{k-1}$ and $u'_i, \ldots, u'_{k-1}$ for $i > k/2$ (receiving $2(k-i)$ links from U); as well as a link from each of $w_{k+1-i}, \ldots, w_{k-1}$ for $i \leqslant k/2$ (meaning u_1 receives no such link) and a link from each of $w_1, \ldots, w_{i-1}$ for $i > k/2$ (receiving $i - 1$ links from W). Then, for all $i < k$, v_i receives two more links from U and one less link from W than

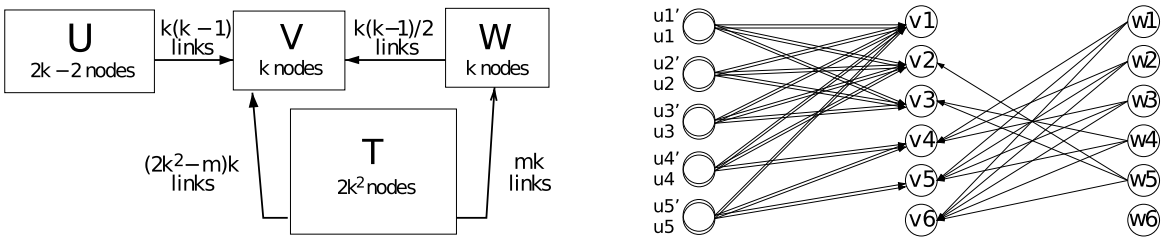

Fig. 1. The graph G (Theorem 1) formed by the 4 blocks of nodes U, V, W and T (left); and the specific link structure of U, V and W for $k = 4$ (right).

v_{i+1}, and $P_{\bar{d}}(v_i) - P_{\bar{d}}(v_{i+1}) \propto (\bar{d}\frac{2}{k/2}) - (\bar{d}\frac{1}{k/2} + \bar{d}^2\frac{m}{k}\frac{1}{k/2}) = \frac{\bar{d}}{k/2}(1 - \bar{d}\frac{m}{k})$. The last term is positive for $\bar{d} = d$ and negative for $\bar{d} = d + 1/k$, proving the thesis. □

Proof of Theorem 2. We (arbitrarily) choose $F = |\Pi|$ distinct values $d_1, \ldots, d_F$ in $\mathcal{I}$, and, for each d_h, a k-tuple of scores $s_{h1}, \ldots, s_{hk}$ for $v_1, \ldots, v_k$ that induces a distinct permutation in Π. For each node v_j we build a system $AX_j = B_j$, where

$$A = \begin{pmatrix} d_1^1 & \ldots & d_1^F \\ & \ldots & \\ d_F^1 & \ldots & d_F^F \end{pmatrix}, \qquad X_j = \begin{pmatrix} x_{1j} \\ \ldots \\ x_{Fj} \end{pmatrix}, \qquad B_j = \begin{pmatrix} s_{1j} \\ \ldots \\ s_{Fj} \end{pmatrix}$$

whose solutions $x_{1j}, \ldots, x_{Fj}$ are the branching contributions at levels $1, \ldots, F$ that the graph must provide to v_j in order to satisfy the score assignment (note that A is non-singular since it is the product of a Vandermonde matrix and a non-singular matrix). Although it might not be possible to construct a graph that provides these exact contributions, we show how to build one that maintains the same ranking of $v_1, \ldots, v_k$ for each value d_h of the damping factor. We use the solutions of the systems to iteratively build an *ancestor forest* of k trees (see Fig. 2), where at step h we add the nodes at depth h in each tree (step 0 adds the k roots $v_1, \ldots, v_k$).

We first assume that $x_{hj} > 0$ for all $j = 1, \ldots, k$, and build level h of the ancestor tree of v_j as follows. Let u be a node in level $h-1$ of the ancestor tree of v_j, and let p be the unique (as will immediately be clear) path from u to v_j. Create n_{hj} new nodes and add them a link to u; then add $\omega_{hj} - 1$ children to each of these new nodes. Note that now there is a unique path from each of these nodes to v_j. Now the branching contribution from the hth level is $(n_{hj}/\omega_{hj})\,\text{branching}(p)$, and we can choose n_{hj}, ω_{hj} such that this contribution approximates x_{hj} to an arbitrary degree of precision. Finally set $\delta_h = 0$ (the reason will be clear soon).

If instead $x_{hj} \leqslant 0$ for some j, let $y_h = min_j\{x_{hj}\}$, and choose $\delta_h > 0$ such that $\delta_h + y_h > 0$. For $j = 1, \ldots, k$ replace x_{hj} with $x_{hj} + \delta_h$ (which is positive by construction) and proceed as above.

We now prove that PageRank ranks $v_1, \ldots, v_k$ in every order in Π as d takes values $d_1, \ldots, d_F$. For simplicity we set the personalization vector $\mathbf{e} = [\frac{1}{n} \ldots \frac{1}{n}]$, but the result can be easily extended to the general case $\mathbf{e} > 0$. Recalling Eq. (1), the PageRank score for v_j is

$$\begin{aligned} P_{v_j}(d_h) &= \frac{1-d_h}{n} \sum_{\ell=0}^{\infty} d_h^\ell \sum_{p \xrightarrow{\ell} v} \text{branching}(p) \\ &= \frac{1-d_h}{n} \sum_{\ell=0}^{F} d_h^\ell \cdot (x_{lj} + \delta_\ell) \\ &= \frac{1-d_h}{n} \sum_{\ell=1}^{F} d_h^\ell \cdot x_{lj} + \frac{1-d_h}{n}\left(1 + \sum_{\ell=0}^{F} d_h^\ell \delta_\ell\right) \\ &= \frac{1-d_h}{n} \cdot s_{hj} + q_h \end{aligned}$$

and thus, for a fixed h, the relative order of $v_1, \ldots, v_k$ follows the choice of the scores $s_{h1}, \ldots, s_{hk}$. To bring $v_1, \ldots, v_k$ to the top k positions, we increase their score by adding m parents to each v_j; i.e., we choose m equal to the maximum size (in nodes) of any ancestor tree. This increases the branching contribution at level 1 of each v_j by the same quantity, but does not affect the ordering; so $v_1, \ldots, v_k$ still assume all possible permutations in Π as d varies in $d_1, \ldots, d_F$.

Some readers might note that — at least in some application domains — one is unlikely to encounter graphs that are acyclic and/or disconnected and wonder about the applicability of Theorem 2 in such cases. It is easy to verify that the PageRank score vector does not change if one adds to every sink of a generic graph links to every other node in the graph, and that this modification makes the graph of Theorem 2 strongly connected. □

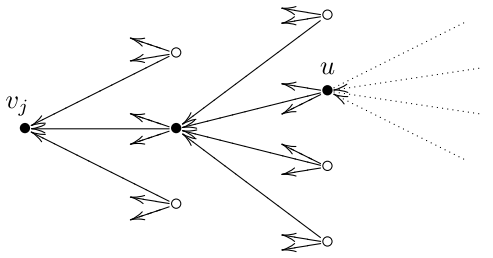

Fig. 2. Ancestor tree for node v_j (Theorem 2). "Empty" nodes are orphans.

3. Lineage analysis

It is natural to ask to what extent PageRank rankings of "real" graphs like the Web graph are affected by variations in the damping factor, particularly in the light of the results of Section 2. Unfortunately, verifying that a node always outranks another as the damping factor varies by discrete increments (e.g. 0.01) *suggests, but cannot conclusively prove* that the first node always outranks the second over a whole continuous interval of variation of the damping factor: by virtue of Theorems 1 and 2 the ranking could drastically change between those isolated sampling points. For the same reason, while previous experimental evidence obtained over a set of isolated values of the damping factor (e.g. [22]) *suggests* that PageRank is indeed stable even for large variations, this evidence cannot be taken as conclusive. This section shows how to bypass this problem.

Just like Shannon's Sampling Theorem proves that a continuous signal can be reconstructed from a finite number of samples so long as it is bandwidth-limited, Section 3.1 shows that, so long as PageRank can be approximated using only a finite number of iterations, one can exploit Sturm's theorem to verify if a node outranks another for all damping factors in a finite number of steps. We also consider a more "robust" notion of rank dominance between nodes — not only for all damping factors, but also for all "damping variables" — informally, time-variant damping factors introduced in Section 3.2 and related to the *damping functions* of [3]. The core result of this section is a simple condition that can be checked efficiently and yet is both necessary and sufficient to guarantee that a node outranks another for all damping variables. Section 3.3 presents this "dominance" condition, which is based on the concept of *lineage* of a node, and turns out to be a very "natural" concept with a strong intuitive justification totally independent of PageRank.

3.1. Better rank for every damping factor?

In theory, one can check if the relative order of two nodes u, v changes as d varies by verifying if the difference between their PageRank series is both positive and negative in the interval $(0, 1)$. If one considers PageRank computations truncated after t iterations for some t (which is always true in practice), this can be done via Sturm's theorem [14].

Let $\bar{x}$ be a generic root of a real polynomial $Q(x)$ with multiplicity m; then $\bar{x}$ is a root of $Q_1(x) = \mathrm{GCD}(Q(x), \frac{d}{dx}Q(X))$ with multiplicity $m-1$. Sturm's theorem [14] allows one to compute the number of real roots of $Q(x)$ of multiplicity $\geqslant 1$ in the interval $(0, 1)$. Applying it to $Q_1(x) = \mathrm{GCD}(Q(x), \frac{d}{dx}Q(x))$, then to $Q_2 = \mathrm{GCD}(Q_1(x), \frac{d}{dx}Q_1(x))$ and so on, one can compute the number of real roots of multiplicity respectively $\geqslant 2$, $\geqslant 3$, etc. and thus the exact number of real roots of multiplicity $1, 2, 3$, etc. It is then immediate to verify that the relative order of u and v changes as d varies over the interval $(0, 1)$ if and only if the difference between their truncated PageRank series $P_u^t(d) - P_v^t(d)$ has roots of odd multiplicity in that interval.

This method has a number of limitations. First, it does not allow an "incremental refinement" of the rank comparison: it is essentially impossible to extend the analysis obtained by truncating the PageRank series after t steps to include the effects of one more step without recomputing everything "from scratch". Second, its computational complexity is considerable, since Sturm's procedure for a single polynomial in general entails $\Omega(t)$ divisions between polynomials of degree $\approx t$. Third, it also requires considerable effort to avoid finite-precision errors, since it can involve a large number of subtractions and thus "catastrophic cancellations" [19,29]. Fourth, it provides no natural interpretation of *why* a node outranks another for every damping factor. Fifth, the underlying model is not robust, as it does not capture the possibility of random surfers whose probability of "pressing the reset button" (one minus the damping factor) varies with time. The following Section 3.2 explores this last issue in greater detail.

3.2. Damping variables

It is natural to extend the stochastic model of PageRank to one where the probability of following one of the current node's outgoing links is not a constant damping factor d, but is instead a *damping variable* $d(\tau)$ that is a function of the number of steps τ taken since the last random jump. E.g. a Web surfer might have a high probability of following a chain of links up to a depth of e.g. 3, and then of "hitting the reset button" with a random jump. In this case $d(\tau)$ would be close to

1 for $\tau \leqslant 3$, but close to 0 for $\tau > 3$ (we formally define $d(0) = 1$). We can generalize the analysis carried out in Section 2.1 to damping variables, and prove that:

Theorem 3. *If $d(\tau) \leqslant (1-\epsilon)$ for some $\epsilon > 0$ and for infinitely many τ, then the score assigned by PageRank to the node v is*:

$$P_v\big(d(\cdot)\big) = \sum_{\ell=0}^{+\infty} J_{d(\cdot)} \prod_{j=0}^{\ell} d(j) \sum_{p \xrightarrow{\ell} v} \text{branching}(p) \cdot \mathbf{e}_{v_\ell} \tag{2}$$

where $J_{d(\cdot)}$ is the limit for $t \to +\infty$ of the probability $J_{d(\cdot)}(t)$ that a random jump occurs at time t, and $J_{d(\cdot)} \prod_{j=0}^{i-1} d(j)$ is the limit for $t \to +\infty$ of the damping function [3] *denoting the probability of having taken the last random jump exactly i steps before time t.*

While the meaning of Eq. (2) and its relationship to Eq. (1) are quite intuitive, proving that it holds requires some subtlety; indeed, if the condition $d(\tau) \leqslant (1-\epsilon)$ is not satisfied (e.g. if one only requires $d(\tau) < 1$) then the probability of being at node v at time t might not become stationary as $t \to \infty$. In this case, one might still *formally define* the score of a node through Eq. (2), but relating it to a stochastic surfing model becomes considerably harder; and, perhaps more importantly, *computing* the score of a node becomes considerably more difficult, since the classic iterative algorithms based on the power method can fail to converge.

Note that PageRank in practice is computed using only a finite number $\bar{\tau}$ of iterations of the power method, effectively truncating the series in Eqs. (1) and (2) to the τth term — i.e. employing, rather than a constant damping factor $\bar{d}$, a damping variable $d(\tau)$ equal to $\bar{d}$ for $\tau \leqslant \bar{\tau}$ and to 0 for $\tau > \bar{\tau}$. This provides further justification for the introduction of damping variables.

The proof that, indeed, the stochastic process above does produce the stationary probability described by Theorem 3 may be skipped without impairing the understanding of the rest of the paper.

Proof of Theorem 3. We first show that the probability $J_{d(\cdot)}(\ell)$ that a random jump occurs at time ℓ converges to a limit. More formally, let $\epsilon > 0$ and let $d : \mathbb{N} \mapsto [0,1]$ such that $d(\tau) \leqslant 1-\epsilon$ for infinitely many τ. Then the following limit exists:

$$J_{d(\cdot)} \triangleq \lim_{\ell \to +\infty} J_{d(\cdot)}(\ell). \tag{3}$$

Let $L(\ell)$ be the time of the last jump before ℓ. By the law of total probability,

$$J_{d(\cdot)}(\ell) = \sum_{i=0}^{\ell-1} P\big(\text{jump at } \ell \;\big|\; L(\ell) = i\big) P(\text{jump at } i) = \sum_{i=0}^{\ell-1} \big(1 - d(\ell-i)\big) \left(\prod_{j=0}^{\ell-i-1} d(j) \right) J_{d(\cdot)}(i) = \sum_{i=0}^{\ell-1} a_{\ell-i} J_{d(\cdot)}(i)$$

where $a_{\ell-i} = (1-d(\ell-i))(\prod_{j=0}^{\ell-i-1} d(j))$. We break up the last summation as $\sum_{i=0}^{\ell-k-1} a_{\ell-i} J_{d(\cdot)}(i) + \sum_{i=\ell-k}^{\ell-1} a_{\ell-i} J_{d(\cdot)}(i)$; we show that the first term vanishes and the whole summation tends to the second term, which converges. The first term satisfies $\sum_{i=0}^{\ell-k-1} a_{\ell-i} J_{d(\cdot)}(i) \leqslant \sum_{i=0}^{\ell-k-1} a_{\ell-i} = \prod_{j=0}^{k} d(j)$, thus we choose a sufficiently large constant k_0 such that, for $\ell > k > k_0$, we have $d(j) \leqslant 1-\epsilon$ for at least m values of j in $0, \ldots, k$, where m is such that $(1-\epsilon)^m < \delta/2$; and then $\prod_{j=0}^{k} d(j) < \delta/2$.

Let us prove that the second term converges for $\ell, k \to \infty$. The last equation can be rewritten as $x(\ell) = \sum_{i=1}^{k} a_i x(\ell-i) + \delta'$, where $\delta' < \delta/2$. The characteristic polynomial is then $\sum_{i=1}^{k} a_i z^i - (1-\delta')$, and the sum of all terms a_i is $1 - \prod_{j=0}^{k} d(j)$; the last quantity is strictly less than $1-\delta'$ for an appropriate choice of k (if $d(j) = 0$ for $j < k$, $k_0 = j-1$ implies $\delta' = 0$). The roots of the polynomial lie outside the unit circle and the system converges to a limit $J_{d(\cdot)}$.

We can now prove the theorem. Choose $\delta > 0$. The probability of being on node v at time ℓ is $P_v^{\ell}(d(\cdot)) = \sum_{i=0}^{\ell} \pi_v(i) J_{d(\cdot)}(\ell-i)$ where $\pi_v(i) = \prod_{j=0}^{i} d(j) \sum_{p \xrightarrow{i} v} \text{branching}(p) \cdot \mathbf{e}_{p_i}$ is the probability of reaching v following i links after a random jump. We rewrite $P_v^{\ell}(d(\cdot)) = \sum_{i=0}^{\lambda} \pi_v(i) J_{d(\cdot)}(\ell-i) + \sum_{i=\lambda+1}^{\ell} \pi_v(i) J_{d(\cdot)}(\ell-i)$ and bound the two terms. By Eq. (3), the ith addend of the first term converges to $J_{d(\cdot)} \pi_v(i)$ as $\ell \to +\infty$ for a fixed λ. Thus we can choose ℓ_0 such that, for every $\ell \geqslant \ell_0$,

$$\left| J_{d(\cdot)} \sum_{i=0}^{\lambda} \pi_v(i) - \sum_{i=0}^{\lambda} \pi_v(i) J_{d(\cdot)}(\ell-i) \right| \leqslant \sum_{i=0}^{\lambda} \big| J_{d(\cdot)} - J_{d(\cdot)}(\ell-i) \big| < \sum_{i=0}^{\lambda} \frac{\delta}{2(\lambda+1)} = \delta/2.$$

The second term is the probability of reaching v at time ℓ having taken the last random jump before time $\ell - \lambda$; therefore it is bounded by the probability of having taken a path of length λ, which is no more than $c(\lambda)$. We can now choose $\lambda \geqslant \lambda_0$ large enough to ensure that $c(\lambda) < (1-\epsilon)^m < \delta/4$.

The two bounds yield, for $\ell \geqslant \ell_0$ and $\lambda > \lambda_0$,

$$\left| J_{d(\cdot)} \sum_{i=0}^{\ell} \pi_v(i) - P_v^{\ell}\big(d(\cdot)\big) \right|$$
$$\leqslant \left| J_{d(\cdot)} \sum_{i=0}^{\lambda} \pi_v(i) - \sum_{i=0}^{\lambda} \pi_v(i) J_{d(\cdot)}(\ell - i) \right| + \left| J_{d(\cdot)} \sum_{i=\lambda+1}^{\ell} \pi_v(i) \right| + \left| \sum_{i=\lambda+1}^{\ell} \pi_v(i) J_{d(\cdot)}(\ell - i) \right| \leqslant \delta.$$

In conclusion, $\lim_{\ell \to +\infty} P_v^{\ell}(d(\cdot)) = J_{d(\cdot)} \sum_{i=0}^{+\infty} \pi_v(i)$. □

For each node v, we now have a well-defined score $P_v(d(\cdot))$. The score vector has a representation which is very similar to that of the PageRank vector:

$$P\big(d(\cdot)\big) = J_{d(\cdot)} \sum_{\ell=0}^{+\infty} c_\ell \mathbf{M}^\ell \cdot \mathbf{e} \tag{4}$$

where $c_\ell = \prod_{j=0}^{\ell} d(j)$. Note that for constant $d(\cdot) = d$ we have $J_{d(\cdot)} = 1 - d$ and $c_\ell = d^\ell$, which yields the classic PageRank formulation as a power series that can be found e.g. in [3].

3.3. Lineages

This subsection provides a simple condition both necessary and sufficient to guarantee that, for all damping variables $d(\cdot)$, the score $P_v(d(\cdot))$ of a node v is always at least as high as the score $P_w(d(\cdot))$ of a node w.

Recall the term $\sum_{p \xrightarrow{\ell} v} \text{branching}(p) \cdot \mathbf{e}_{v_\ell}$ in Eq. (2) — the level ℓ branching contribution to the score of v. Informally, the mth generation of the *lineage* of v is equal to the sum of the branching contributions of all levels $\ell \leqslant m$. More precisely:

Definition 1. The mth generation of the lineage of v is $L_v(m) = \sum_{\ell=0}^{m} \sum_{p \xrightarrow{\ell} v} \text{branching}(p) \cdot \mathbf{e}_{v_\ell}$.

We say that the lineage $L_v(\cdot)$ of a node v is greater than or equal to the lineage $L_w(\cdot)$ of a node w if it is greater or equal at every generation, in which case we write simply $L_v \geqslant L_w$.

There is a simple, intuitive interpretation of dominance between lineages. If we imagine that the authority/reputation/trust of a node is divided evenly among the nodes it points to, having a greater lineage at the mth generation means receiving more authority from all nodes within at most m hops. Note that in such a comparison nodes that are further than m hops away are completely disregarded: this models the fact that, after all, authority/reputation/trust may be inherited, but only to a point. If one is uncertain to *which* point, one can be sure that a node has authority at least as high as that of another node only if its lineage is at least as high at *every generation*.

And, indeed, we show that having a lineage that is at least as high at every generation is strictly equivalent to receiving an equal or higher score by PageRank for every damping variable. More formally, we prove:

Theorem 4. $L_v \geqslant L_w \Leftrightarrow \forall d(\cdot) \in (0,1), P_v(d(\cdot)) \geqslant P_w(d(\cdot))$.

Proof. We first prove the ($\Rightarrow$) side. Recall $c_i = \prod_{j=0}^{i} d(j)$. The score vector is a convex linear combination of lineages:

$$P_v\big(d(\cdot)\big) = J_{d(\cdot)} \sum_{i=0}^{+\infty} b_i L_v(i) \tag{5}$$

where $b_i = c_i - c_{i+1} \geqslant 0$. Then $P_v(d(\cdot)) \geqslant P_w(d(\cdot))$.

We now prove the ($\Leftarrow$) side. For a generic $k \geqslant 0$, and a generic $\epsilon > 0$, consider the function:

$$d(i) = \begin{cases} 1 - \epsilon, & i = 0, \ldots, k, \\ \epsilon, & i > k \end{cases}$$

where $\epsilon \in (0,1)$ will be defined later. Let $P^{(k)}(d(\cdot))$ be equal to $J_{d(\cdot)} \sum_{j=0}^{k} c_j \mathbf{M}^j \cdot \mathbf{e}$, i.e. to the truncation of the PageRank series to the term of order k. Then we have:

$$P_v\big(d(\cdot)\big) = J_{d(\cdot)} \sum_{i=0}^{k} c_i \big(\mathbf{M}^i \cdot \mathbf{e}\big)_v + J_{d(\cdot)} \sum_{i=k+1}^{\infty} c_i \big(\mathbf{M}^i \cdot \mathbf{e}\big)_v$$
$$\leqslant P_v^{(k)}\big(d(\cdot)\big) + J_{d(\cdot)} \sum_{i=k+1}^{\infty} (1-\epsilon)^k \epsilon^{i-k} n$$
$$\leqslant P_v^{(k)}\big(d(\cdot)\big) + J_{d(\cdot)} \epsilon (1-\epsilon)^{k-1} n.$$

Considering nodes v and w we have:

$$P_v^{(k)}\big(d(\cdot)\big) + J_{d(\cdot)}\epsilon(1-\epsilon)^{k-1}n \geqslant P_v\big(d(\cdot)\big) \geqslant P_w\big(d(\cdot)\big) \geqslant P_w^{(k)}\big(d(\cdot)\big)$$

which can be rewritten as

$$(1-\epsilon)^k L_v(k) + \epsilon\sum_{i=0}^{k-1}(1-\epsilon)^i L_v(i) + \epsilon J_{d(\cdot)}(1-\epsilon)^{k-1}n \geqslant (1-\epsilon)^k L_w(k) + \epsilon\sum_{i=0}^{k-1}(1-\epsilon)^i L_w(i).$$

Since the last equation must hold for every (arbitrarily small) $\epsilon > 0$, by continuity $L_v(k) \geqslant L_w(k)$. □

Note that in practice we do not need to check lineage dominance at *every generation*; if we restrict ourselves to damping variables that are 0 for $\tau > t$, we only need to check at most t generations. This is equivalent to considering PageRank computations truncated after at most t iterations (effectively truncating the series in Eqs. (2) and (4) to the tth term), which is always true in practice. Thus, one can check if the relative order of two nodes is the same for every damping variable in $O(t)$ comparisons; by contrast, the algorithm based on Sturm's theorem described in Section 3.1 requires $\Omega(t^2)$ operations.

In addition, comparing lineages yields computations that have considerably fewer problems of numerical stability than those involving Sturm chains. While an in-depth exploration of this issue is beyond the scope of this paper (see e.g. [30]), it is immediate to verify that computing the ℓth lineage involves summations where each term is (a) positive and (b) obtained as the inverse of a product of a number of integers equal to the lineage. The products can be computed with a loss of accuracy equal to at most $\lceil \log_2(\ell) \rceil$ bits; the sum entails an additional loss of at most 1 bit of accuracy (see e.g. [19,29]).

Thus, on any graph of size n and outdegree at most g on which PageRank converges in at most ℓ iterations, a floating point arithmetic with exponents of $\max(\log_2\log_2(\ell n), \log_2\log_2(g^\ell))$ bits and significands greater than $\approx \log_2(1/\epsilon) + \log_2(\ell)$ bits suffices to discriminate between lineages within a factor $1+\epsilon$ of each other. In particular, IEEE 754 double precision arithmetic appears more than sufficient to guarantee several dozen bits of precision on the Web graph and on any "lesser" graph.

4. Damping-independent ranking

Lineages and Theorem 4 provide powerful tools to compare two nodes over the spectrum of all damping variables. This section leverages them to introduce the concepts of *strong rank* and *weak rank* (Section 4.1), and the related ranking algorithms *StrongRank* and *WeakRank* (Section 4.2). These provide interesting measures of the "fuzziness" of rankings, of the "orderability" of the nodes of a graph, and of the performance of different ranking algorithms based on the random surfer model.

4.1. Strong and weak rank

Given a node v, and assuming for simplicity all ties in lineage score are broken (e.g. arbitrarily), all other nodes fall into three sets: the set $S(v)$ of nodes *stronger* than v (with a greater lineage), the set $W(v)$ of nodes *weaker* than v (with a lesser lineage), and the set $I(v)$ of nodes *incomparable* with v (with a lineage greater at some generations and lesser at others). The cardinalities of these sets define the *weak* and *strong* rank of v:

Definition 2. The *weak rank* $\rho_w(v)$ and the *strong rank* $\rho_s(v)$ of a node v are, respectively, $|S(v)|+1$ and $|S(v)\cup I(v)|+1$.

Note that $\rho_w(v)-1$ is the number of those nodes that outperform v for all damping variables, whereas $\rho_s(v)-1$ is the number of those nodes which outperform v for at least one damping variable. Thus $\rho_w(v)$ is a lower bound to the minimum (i.e. best) rank achievable by v, while $\rho_s(v)$ is an upper bound to the maximum (i.e. worse) rank achievable by v; and $\rho_s(v)-\rho_w(v)$ is then an upper bound to the maximum variation in v's rank. Note that any or all of these three bounds might hold strictly, and there is a subtle but profound difference between $\rho_s(v)-\rho_w(v)$ and the maximum variation of v's rank that makes the former more descriptive of the "fuzziness" of v's performance. E.g. suppose that v holds 10th rank for all damping variables, but 999 other nodes fill the top 9 positions in turn for different damping variables. In this case $\rho_s(v) = 1000$, since v does not fare definitively better than any of those 999 nodes; and $\rho_w(v) = 1$, since none of those 999 nodes fares definitively better than v.

Strong and weak rank also provide a measure of the *global* rank fuzziness in a generic graph — that is, of the extent to which the graph can be ordered satisfying simultaneously every type of user (with different user behaviours described by different damping variables). For a graph G, consider the number $s_k(G)$ of nodes with strong rank k or less. If $s_k(G) \approx k$, then every user's top k set has a high density of nodes also in the top k set of *every other* user. Conversely, if $s_k(G) \ll k$, then few nodes will be of "universal importance". Similarly, consider the number $w_k(G)$ of nodes with weak rank k or less. If $w_k(G) \approx k$, then relatively few nodes are sufficient to include the k most important nodes of *every other* user. Conversely, if $w_k(G) \gg k$, then the behaviours of different users are sufficiently diverse that, in order to ensure that no user misses the items of the greatest importance to him, any algorithm must return a very large collection of items.

The ratio $w_k(G)/s_k(G)$ can then be seen as a measure of the *inevitable* price of obliviousness to the user's model: the smaller it is, the more well-orderable G is. In the ideal case, $s_k(G) = k = w_k(G)$, all users have exactly the same preferences and every damping variable yields the same ordering.

4.2. StrongRank and WeakRank

Strong rank and weak rank automatically induce two new ranking algorithms, *StrongRank* and *WeakRank*, that rank nodes respectively in order of strong and weak rank. Neither necessarily corresponds to PageRank for some damping variable. StrongRank tends to return items that are each of at least moderate importance for every user (with different user behaviours described by different damping variables). WeakRank tends to return, for every user, at least a few items that are of high importance for that user. The evaluation of the effectiveness of StrongRank and WeakRank as practical ranking algorithms (either on the Web or in other application domains) is certainly a promising direction of future research.

It is immediately obvious, however, that StrongRank and WeakRank can provide benchmarks to classify the performance of other ranking algorithms based on the surfer model. If for every k many of the top k items returned by a ranking algorithm are also among the top k items returned by StrongRank (the "intersection metric" of [11]), one can reasonably deduce that a large fraction of items returned by that algorithm is of at least moderate importance for every user. Similarly, a large intersection with WeakRank points to an algorithm that returns, for every user, a large fraction of that user's top choices.

A complete analysis of the complexity of StrongRank and WeakRank would require taking into account the properties of the target graph (and thus of the application domain) as well as caching and parallelizability issues. This is beyond the scope of this paper; however, the remainder of this section provides a few basic results (that can be skipped without impairing the understanding of the rest of the paper).

It is not difficult to prove that the *worst case* complexity of PageRank (for *one* value of the damping factor) on a graph of n nodes equals that of StrongRank and WeakRank:

Theorem 5. *The worst case complexity of computing StrongRank and WeakRank up to lineage ℓ on an n node graph is $O(\ell n^2)$, equal to that of computing the first ℓ iterations of PageRank.*

StrongRank and WeakRank can then be computed for any graph of up to millions of nodes on a PC in a few days; and very few application domains of PageRank entail larger graphs (the World Wide Web being one notable exception). Graphs of much larger size n are manageable by PageRank itself only if of low (average) degree $g \ll n$; and in such graphs, one is often interested only in the top k ranking nodes, with $k \ll n$. When these graphs are Δ-*well orderable*, i.e. there are at most $k\Delta$ nodes with weak rank less than k and at least k/Δ with strong rank less than k (this seems to hold with $2 \leqslant \Delta \leqslant 4$ for $k > 100$ in social networks like the Web, see Section 5) we can refine Theorem 5 into Theorem 6:

Theorem 6. *The worst case complexity of computing the top k ranks of StrongRank and WeakRank up to lineage ℓ on a Δ-well orderable graph of average degree g and n nodes is, respectively, $O(n\ell g(1 + \frac{\log(k)}{g} + \frac{\Delta(k+\ell)(\log(\Delta)+k)}{ng}))$ and $O(n\ell g(1 + \frac{\log(k)}{g} + \frac{k^2\Delta^3+\Delta(k+\ell)(\log(\Delta)+k)}{ng}))$ vs. a complexity of $\Theta(n\ell g)$ for the top k ranks and the first ℓ iterations of PageRank.*

For $\max(\log(n), g, \ell) \leqslant k \leqslant \sqrt{n}$ the complexity of StrongRank and WeakRank becomes respectively $O(n\ell g(1 + \frac{\log(k)+\Delta}{g}))$ and $O(n\ell g(1 + \frac{\log(k)+\Delta^3}{g}))$. Thus, even in the case of the (indexed) Web graph it still appears possible to compute the top $\approx 10^5$ ranks of StrongRank and WeakRank in time comparable to that of PageRank (within an order of magnitude — several hours on a single PC).

The rest of this section is devoted to the proof of Theorem 6, of which Theorem 5 is a corollary.

Proof. We first describe two initial "preprocessing" steps that are common to StrongRank and WeakRank, before detailing how each algorithm computes its top k ranked nodes; finally, we analyse the worst case complexity for the top k ranks and the first ℓ iterations of PageRank.

The first preprocessing step computes the lineages, up to level ℓ, of all the n nodes of the graph. This can be done in time $O(ngl)$ by iteratively computing the lineage at each level i, which takes an average $O(g)$ (the average (in)degree of the graph) for each of the n nodes. The second preprocessing step extracts, for each of the ℓ levels, the top k nodes (as ranked by lineage) in non-decreasing order. This takes $O(n\log(k))$ steps for each level using a max-heap of size k. The cost of performing these two steps adds up to $O(\ln g + \ln\log(k))$.

To find the top k nodes ranked by StrongRank, it is sufficient to consider all the nodes with strong rank less than $k\Delta$, which by definition of Δ are at least k. These nodes belong necessarily to the intersection of the sets of top $k\Delta$ nodes (as ranked by lineage) at each level. Since Δ is unknown in advance, one can use a binary search over the size of these sets until an intersection of size between k and $k + \ell$ is found; this takes $O(\log(k\Delta))$ steps, each computing the intersection between ℓ sets of size $O((k + \ell)\Delta)$ each; and the cost sums up to $O(\log(k\Delta)\ell(k + \ell)\Delta)$. For each of the (less than) $k + \ell$

nodes in the intersection found, one computes its strong rank as the number of nodes which are ranked higher (by lineage) in at least one level. This takes $O(kl\Delta)$ steps for each node, for a total of $O((k+\ell)kl\Delta)$ steps. Thus the cost of StrongRank, including the two preprocessing steps, is $O(\ln g + \ln\log(k) + \log(k\Delta)\ell(k+\ell)\Delta + (k+\ell)lk\Delta)$. When k is of reasonable size (i.e. $\log(k) = O(g)$) and when, as in the case of the Web graph, Δ is a small constant and $\ell = O(k)$ (i.e. PageRank converges in a few thousand iterations) then, in the formula above, the second term is $O(\ln g)$, the third term is $O(\ell k\Delta g) = O(\ln g)$ (since $k\Delta \leqslant n$), the fourth term is $O(\ell k^2\Delta) = O(\ell n\Delta)$. This yields a total cost of $O(\ln g + \ln\Delta)$ steps.

To find the top k nodes ranked by WeakRank, it is sufficient to consider all nodes with weak rank less than k, which are at least k and, by definition of Δ, no more than $k\Delta$. These nodes belong necessarily to the union of the sets of top $k\Delta$ nodes (as ranked by lineage) at each level; i.e. to the union of the same sets considered in the computation of StrongRank above, yielding the same cost. This union has cardinality at most $k\Delta^2$ — otherwise there would be more than $(k\Delta)\Delta$ nodes with weak rank less than $k\Delta$, which is impossible by definition of Δ. For each of these $O(k\Delta^2)$ nodes, its weak rank is the number of nodes ranked higher (by lineage) at each level. This (using lineage rankings) takes no more than $kl\Delta$ steps for each node, and $O(k^2\ell\Delta^3)$ total steps. Thus the cost of WeakRank, including the two preprocessing steps, is $O(\ln g + \ln\log(k) + \log(k\Delta)\ell(k+\ell)\Delta + k^2\ell\Delta^3)$; under the same assumptions made above, this becomes $O(\ln g + \ln\Delta^3)$.

Note that the cost of computing the top k nodes ranked by PageRank using the first ℓ iterations is asymptotically no less that the cost of the first preprocessing step of StrongRank and WeakRank, $\Omega(\ln g)$, plus the cost of extracting the top k nodes, which is $\Omega(n\log(k))$. This yields a total cost of $\Omega(\ln g)$ steps for PageRank. □

In the Web graph Δ appears to be bounded by a small constant not exceeding 2.5 (see Section 5.2 below). Thus we could run StrongRank and WeakRank for $k = 20\,000$ even for a fairly large snapshot of the .it domain in a few hours on a personal computer.

5. Experimental results

This section brings the analytical machinery of Section 3 to bear on "real" graphs — a 2004 snapshot of the .it domain Web graph, and a 2007 snapshot of the CiteSeer citation graph. Section 5.1 briefly presents the data and the experimental setup. Section 5.2 evaluates the extent to which the nodes of those graphs can be ordered in a fashion satisfying simultaneously every user (with different user behaviours described by different damping variables). Finally, Section 5.3 analyses the performance of PageRank as d varies from nearly 0 to nearly 1 by evaluating the intersection of its top k node set with the top k node set of StrongRank and WeakRank.

5.1. Data and experimental setup

We analyse the $40M$ node, $1G$ link snapshot of the 2004 .it domain published by [27], and the $0.7M$ node, $1.7M$ link snapshot of the late 2007 CiteSeer citation graph [9] (nodes are articles and links are citations). We use the WebGraph package [6,27] for their manipulation.

.it, as a national first level domain, provides a large and (unlike e.g. .com or .gov) "well-rounded" portion of the Web graph with a size still within reach of our storage and computational resources. Furthermore, the primary language of .it is not shared by any other country (unlike e.g. .uk or .fr), minimizing distortions in ranking introduced by the inevitable cut of links with the rest of the Web. We perform two pre-processing steps on the .it snapshot. First, we remove intra-domain links (which constitute strongly biased conferral of authority, see [18] and [15]). Second, we merge all dynamic pages generated from the same base page, dealing with pages (like the homepage of a forum's dynamic language) automatically linked by a huge collection of other template-generated pages. Both steps appear to markedly improve the human-perceived quality of the results.

Then, we compute the lineage of each node up to the 128th generation for both graphs. Note that, in theory, lineages should be compared for *all* generations. Stopping at the 128th generation is equivalent to considering damping variables $d(\tau)$ that are 0 for $\tau > 128$, and (typical) PageRank implementations that compute the score vector using at most 128 iterations and disregarding authority propagation over paths longer than 128 hops. This appears reasonable because for all damping factors/variables except those extremely close to 1, the branching contribution of levels beyond the 128th is dwarfed by rounding errors of finite precision computing machinery; and because, empirically, the set of the top k items returned by StrongRank and WeakRank when considering only the first ℓ generations appears to almost completely stabilize as ℓ grows larger than 60–100 (Figs. 3 and 4 show the normalized intersection between the top k item sets returned by StrongRank and WeakRank when considering 128 lineages, and the top k item sets returned when considering ℓ lineages for $\ell = 1, \ldots, 128$).

5.2. Choose the damping, choose the ranking?

Figs. 5 and 6 show the number of k-strongly ranked nodes and the number of k-weakly ranked nodes for the .it domain graph and the CiteSeer citation graph as k varies from 1 to 16 000. The two graphs exhibit a strikingly similar behaviour, with a few differences.

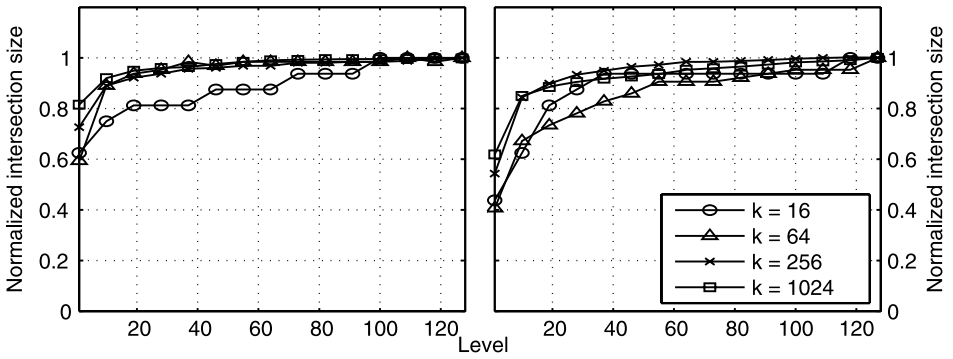

Fig. 3. .it Web graph: convergence of top k-strong (left) and top k-weak (right) sets as a function of lineage generation, for different values of k.

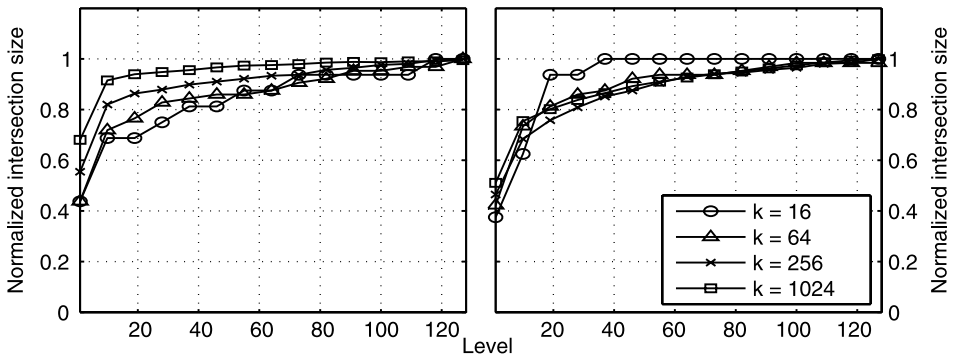

Fig. 4. CiteSeer citation graph: convergence of top k-strong (left) and top k-weak (right) sets as a function of lineage generation, for different values of k.

Both in the .it and in the CiteSeer graph the ratio w_k/k between the number w_k of k-weakly ranked nodes and k is never higher than 6, and converges relatively quickly to a value between 1.5 and 2.5 (it is never larger than 4 for $k \geqslant 7$ on the .it graph, and for $k \geqslant 127$ on the CiteSeer graph). This ratio represents the *inevitable cost of obliviousness to the user model*: the set of all those items that are outranked by less than k other items for some damping variable has size at least w_k.

The ratio s_k/k between the number s_k of k-strongly ranked nodes and k is considerably smaller than 1 in both graphs — particularly in the CiteSeer graph. This ratio represents the fraction of items that are *robustly* in the top k, for all damping factors and variables. For the CiteSeer graph, the ratio is ≈ 0.05 for $k < 200$, converging to a value between 0.5 and 0.6 for $k > 10\,000$. For the .it graph, it is slightly larger: between 0.1 and 0.2 for $k < 200$, converging to a value between 0.4 and 0.5 for $k > 10\,000$.

Thus, ranking sensitivity to the damping factor in "real" graphs appears not nearly as high as that of the synthetic graphs of Section 2, but still considerable — particularly for the top 10–100 items. To return all items that would appear among the top k for *some* damping factor or variable, even the "best" ranking algorithm might have to return from 2 to 4 times as many items; and although *any* choice of the damping factor will guarantee among the top k a non-negligible core of items that would also be returned among the top k for every other choice, this core appears relatively small, between 5% and 40% of the total.

5.3. The best damping factor

In this subsection we deploy StrongRank and WeakRank as a testbed to evaluate the performance of PageRank for different damping factors (see Section 3.3) on the .it graph and on the CiteSeer graph. Figs. 7, 8, 9 and 10 show the fraction of the top k items returned by PageRank that are also among the top k items returned respectively by StrongRank and WeakRank, for $k = 16$, 64, 256 and 1024, as the damping factor varies between 0.01 and 0.99 in steps of 0.01.

Again, the behaviour of PageRank is strikingly similar on the two graphs. For both of them and for all four values of k, the size of the intersection set is between $0.38k$ and $0.93k$ for all damping factors sampled (but note that, in theory, anything could happen between those discrete sampling points — see Section 2!). Thus, for all damping factors, including ones very close to 0 that make the computation of the score vector particularly efficient [5], PageRank always returns a set of results in common with both StrongRank and WeakRank of reasonable size.

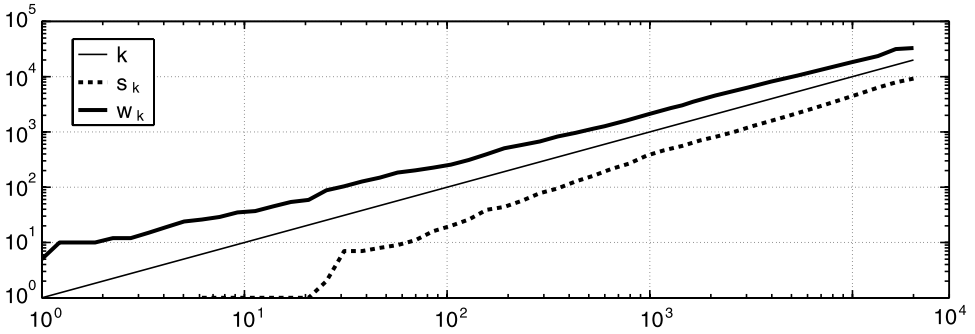

Fig. 5. .it Web graph: s_k (number of k-strongly ranked nodes) and w_k (number of k-weakly ranked nodes), as a function of k.

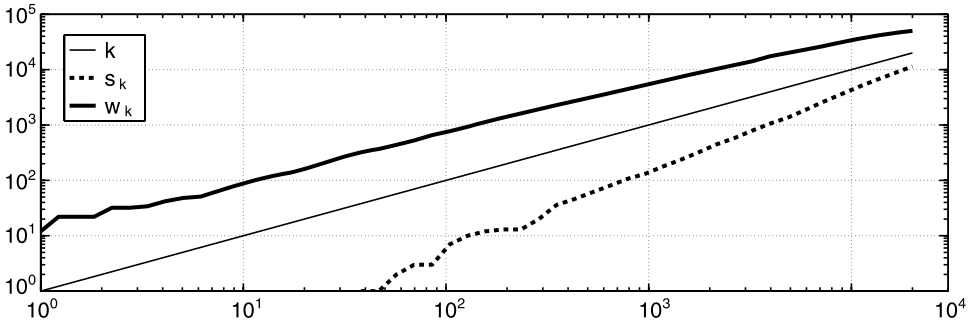

Fig. 6. CiteSeer citation graph: s_k (number of k-strongly ranked nodes) and w_k (number of k-weakly ranked nodes), as a function of k.

For both graphs, the largest intersection with WeakRank is achieved for damping factors in the 0.8–0.9 range; whereas the largest intersection with StrongRank is achieved for damping factors in the 0.5–0.6 range. Which is more desirable? Ultimately, it depends on the nature of the application.

For example, a Web search engine typically returns to a user many more "leads" than that user will follow. In this context a false negative (not returning a "good" page) is far more damaging than a false positive (returning a "mediocre" page). Thus, a large intersection with WeakRank is more desirable, being indicative of an algorithm that returns, for every user, a large fraction of that user's top choices. It is then perhaps not surprising that the typical value assigned to the damping factor in search engines is indeed 0.85!

On the other hand, in a trust/reputation system a false positive (returning as highly trusted an item the user would not trust) is far more damaging than a false negative (not returning as trusted an item the user would actually trust). A large intersection with StrongRank is then more desirable, being indicative of an algorithm that returns only items that are at least moderately trusted by *every* user model; and a damping factor closer to 0.5 — as suggested in [1] — might be a better choice.

6. Conclusions

This paper addresses the fundamental question of how variations in PageRank's damping factor can affect the ranking of nodes. We show that for any k, at least on some graphs, *arbitrarily small* variations in the damping factor can completely reverse the ranking of the top k nodes, or indeed make them assume all possible $k!$ orderings. This is deeply unsatisfying should one be aiming at an "objective" ranking, since an item may be ranked higher or lower than another depending on a tiny variation in the value of a parameter in the user model (the damping factor) likely to differ between different users and, in any case, hard to assess precisely.

Previous experiments that sampled ranking for a handful of different damping factors [19, 11] *suggested* this was not the case at least for the Web graph (leaving the issue open for other application domains). However, verifying rank stability for discrete variations in the damping factor (e.g. 0.01 increments) is not sufficient to conclude that rank is stable as the damping factor varies over a *whole continuous interval* — just like a set of discrete samples from a continuous signal is not, in general, sufficient to reconstruct it. Yet, just like Shannon's Sampling Theorem allows the reconstruction of a continuous signal from a finite set of samples so long as it is limited in bandwidth, *lineage analysis* allows us to compare the rank of two nodes "in one shot" for *all* (even "time variant") damping factors using just a finite set of lineage measurements, so long as PageRank scores can be computed or sufficiently approximated with a limited number of iterations of the Power Method (which is the case for all practical applications).

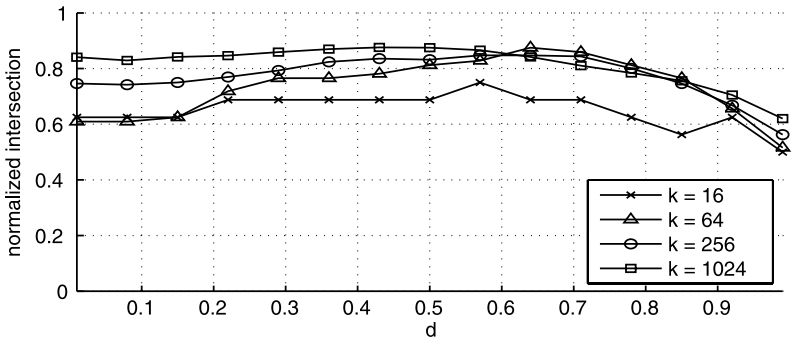

Fig. 7. .it graph: PageRank/StrongRank intersection metric as d varies.

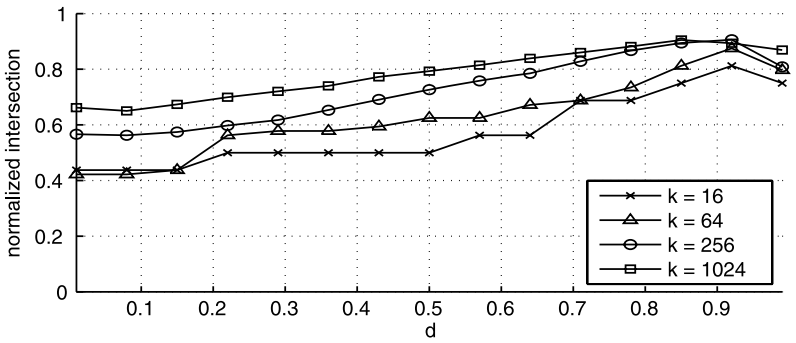

Fig. 8. .it graph: PageRank/WeakRank intersection metric as d varies.

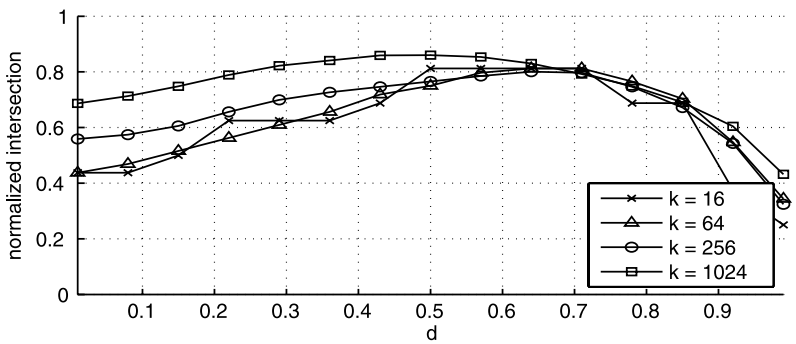

Fig. 9. CiteSeer: PageRank/StrongRank intersection metric as d varies.

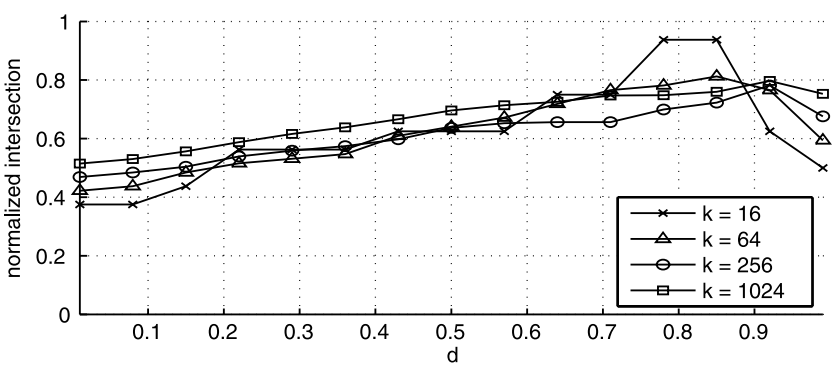

Fig. 10. CiteSeer: PageRank/WeakRank intersection metric as d varies.

Lineage analysis not only allows us to efficiently verify that, indeed, in "real" graphs like Web graphs or citation graphs PageRank is not excessively ranking sensitive (but not insensitive either!) to variations of the damping factor. It also provides a simple, very "natural" interpretation of rank dominance for all damping factors that is completely independent of PageRank. And it allows one to introduce the notions of *strong rank* and *weak rank* of a node, related to, but subtly different from, those of best and worse rank — and more descriptive, since they capture the level of churn "around" a node whose rank is relatively stable overall yet highly unstable compared to its individual competitors.

Weak and strong rank also induce two new ranking algorithms, StrongRank and WeakRank. StrongRank tends to return results that are at least moderately useful under all user models (i.e. for all damping variables). WeakRank tends to return results that are highly useful to at least a niche of users. As such, they can provide useful benchmarks to compare different link analysis algorithms in terms of their ability to, respectively, return "universally useful" results, and to ferret out results of high importance to niches of users. A more thorough evaluation of StrongRank and WeakRank is certainly a promising direction for future research.

StrongRank and WeakRank can also be used to evaluate different values of the damping factor — validating the "folk lore" result that 0.85 is ideal, at least for search engines and other applications where it is far more important to avoid false negatives then false positives. For applications where the reverse is true, our results suggest that a value closer to 0.5 might be more effective, reflecting the results of [1]. It is surprising that, in this regard, the .it and CiteSeer graphs exhibit exactly the same characteristics — it would be interesting to investigate if (and perhaps why) this is the case in other application domains of PageRank.

Acknowledgements

This work was supported in part by EU under Integrated Project AEOLUS (IP-FP6-015964) and in part by Univ. Padova under Strategic Project AACSE. The authors would like to thank Luca Pretto for introducing them to this problem and for his invaluable and unrelenting encouragement and feedback, and the anonymous referees for their constructive criticisms and suggestions.

References

[1] K. Avrachenkov, N. Litvak, K. Son Pham, A singular perturbation approach for choosing PageRank damping factor, ArXiv Mathematics e-prints arXiv:math/0612079.
[2] M. Bacchin, N. Ferro, M. Melucci, The effectiveness of a graph-based algorithm for stemming, in: Proceedings of International Conference on Asian Digital Libraries (ICADL), 2002.
[3] R. Baeza-Yates, P. Boldi, C. Castillo, Generalizing PageRank: Damping functions for link-based ranking algorithms, in: Proceedings of the ACM Special Interest Group on Information Retrieval (SIGIR) Conference, 2006.
[4] M.W. Berry, Survey of Text Mining, Springer-Verlag, 2003.
[5] P. Boldi, M. Santini, S. Vigna, PageRank as a function of the damping factor, in: Proceedings of the ACM World Wide Web Conference (WWW), 2005.
[6] P. Boldi, S. Vigna, The WebGraph framework I: Compression techniques, in: Proceedings of the ACM World Wide Web Conference (WWW), 2004.
[7] S. Chakrabarti, B.E. Dom, D. Gibson, R. Kumar, P. Raghavan, S. Rajagopalan, A. Tomkins, Experiments in topic distillation, in: Proceedings of the ACM Special Interest Group on Information Retrieval (SIGIR) Workshop on Hypertext IR on the Web, 1998.
[8] J. Cho, H. García-Molina, L. Page, Efficient crawling through URL ordering, Computer Networks and ISDN Systems 30 (1–7) (1998) 161–172.
[9] CiteSeer metadata, http://citeseer.ist.psu.edu/oai.html.
[10] G. Erkan, D.R. Radev, LexRank: Graph-based lexical centrality as salience in text summarization, Journal of Artificial Intelligence Research (JAIR) 22 (2004) 457–479.
[11] R. Fagin, R. Kumar, D. Sivakumar, Comparing top k lists, in: Proceedings of the ACM–SIAM Symposium on Discrete Algorithms (SODA), 2003.
[12] D. Gleich, P.W. Glynn, G.H. Golub, C. Greif, Three results on the PageRank vector: Eigenstructure, sensitivity, and the derivative, in: Web Information Retrieval and Linear Algebra Algorithms, 2007.
[13] T.H. Haveliwala, Efficient computation of PageRank, Technical report, 1999.
[14] D.G. Hook, P.R. McAree, Using Sturm Sequences to Bracket Real Roots of Polynomial Equations, Academic Press Professional, Inc., San Diego, CA, USA, 1990.
[15] X.M. Jiang, G.R. Xue, H.J. Zeng, Z. Chen, W.-G. Song, W.-Y. Ma, Exploiting PageRank at different block level, in: Proceedings of the Web Information Systems Engineering (WISE), 2004.
[16] S.D. Kamvar, T.H. Haveliwala, C.D. Manning, G.H. Golub, Extrapolation methods for accelerating PageRank computations, in: Proceedings of the ACM World Wide Web Conference (WWW), 2003.
[17] S.D. Kamvar, M.T. Schlosser, H. Garcia-Molina, The EigenTrust algorithm for reputation management in P2P networks, in: Proceedings of the ACM World Wide Web Conference (WWW), 2003.
[18] J.M. Kleinberg, Authoritative sources in a hyperlinked environment, Journal of the ACM 46 (5) (1999) 604–632.
[19] D.E. Knuth, Art of Computer Programming, vol. 2: Seminumerical Algorithms, 3rd edition, Addison–Wesley Professional, 1997.
[20] A.N. Langville, C.D. Meyer, Deeper inside PageRank, Internet Mathematics 1 (3) (2004) 335–380.
[21] A.N. Langville, C.D. Meyer, Google's PageRank and Beyond: The Science of Search Engine Rankings, Princeton University Press, 2006.
[22] M. Melucci, L. Pretto, PageRank: When order changes, in: Proceedings of the European Conference on Information Retrieval (ECIR), 2007.
[23] L. Page, S. Brin, R. Motwani, T. Winograd, The PageRank citation ranking: Bringing order to the Web, Technical report, Stanford Digital Library Technologies Project, 1998.
[24] E. Peserico, L. Pretto, What does it mean to converge in rank?, in: Proceedings of the International Conference on the Theory of Information Retrieval (ICTIR), 2007.
[25] L. Pretto, A theoretical analysis of Google's PageRank, in: Proceedings of the International Symposium on String Processing and Information Retrieval (SPIRE), 2002.

[26] P. Tarau, R. Mihalcea, E. Figa, Semantic document engineering with WordNet and PageRank, in: Proceedings of the ACM Symposium on Applied Computing (SAC), 2005.
[27] University Milano, Laboratory for Web Algorithmics, http://law.dsi.unimi.it/.
[28] K.W. Wangk, Item selection by 'hub-authority' profit ranking, in: Proceedings of the ACM Special Interest Group on Knowledge Discovery and Data Mining (SIGKDD) Conference, 2002.
[29] J.H. Wilkinson, Rounding Errors in Algebraic Processes, Dover Publications, Inc., 1994.
[30] R. Wills, I. Ipsen, Ordinal ranking for Google's PageRank, SIAM Journal on Matrix Analysis and Applications (SIMAX) 30 (4) (2009) 1677–1696.