Internet Mathematics, 12:68–84, 2016

ISSN: 1542-7951 print/1944-9488 online
DOI: 10.1080/15427951.2015.1098756

Taylor & Francis
Taylor & Francis Group

LiveRank: HOW TO REFRESH OLD DATASETS

The Dang Huynh,[1] Fabien Mathieu,[1] and Laurent Viennot[2]
[1]*Alcatel-Lucent Bell Labs, France*
[2]*Inria - Université Paris Diderot, Paris, France*

Abstract *This article considers the problem of refreshing a dataset. More precisely, given a collection of nodes gathered at some time (webpages, users from an online social network) along with some structure (hyperlinks, social relationships), we want to identify a significant fraction of the nodes that still exist at present time. The liveness of an old node can be tested through an online query at present time. We call LiveRank a ranking of the old pages so that active nodes are more likely to appear first. The quality of a LiveRank is measured by the number of queries necessary to identify a given fraction of the active nodes when using the LiveRank order. We study different scenarios from a static setting where the LiveRank is computed before any query is made, to dynamic settings where the LiveRank can be updated as queries are processed. Our results show that building on the PageRank can lead to efficient LiveRanks, for web graphs as well as for online social networks.*

1. INTRODUCTION

One of the main challenges for large networks data mining is dealing with the high dynamics of huge datasets; not only are these datasets difficult to gather, but they tend to become obsolete very quickly.

In this article, we are interested in the evolution at a large time scale of any large corpus available online. Our primary focus will be the Web, but our approach encompasses any online data with similar linkage enabling crawling, such as P2P networks or online social networks. We thus focus on batch crawling, in which, starting from a completely outdated snapshot of a large graph such as the Web, we want to identify a significant fraction of the nodes that are still alive now. The interest is twofold.

First, many old snapshots of large graphs are available today. Reconstructing roughly what remains from such archives could result in interesting studies of the long-term evolution of these graphs. For large archives, from which one is interested in a particular type of pages, recrawling the full set of pages can be prohibitive. We propose a procedure to identify as quickly as possible a significant fraction of the still-alive pages. Further selection can then be made to identify a set of pages suitable for the study and then to crawl them. Such techniques would be especially interesting when testing the liveness of an item is

A preliminary version of this work was presented at the 11th Workshop on Algorithms and Models for the Web Graph (WAW '14), Beijing, China, December 2014.

Address correspondence to Fabien Mathieu, Lincs, 23 avenue d'Italie, 75013 Paris, France. E-mail: fabien.mathieu@normalesup.org

Color versions of one or more of the figures in the article can be found online at www.tandfonline.com/uinm.

much lighter than downloading it completely. This is, for instance, the case for the Web with HEAD queries compared to GET queries. If a large amount of work has been devoted to maintaining a fresh set of crawled pages, little attention has been paid to the coverage of a partial recrawling of a fairly old snapshot.

Second, some graphs tend to be harder to crawl with time. For instance, Twitter has continuously restricted its capacity to be crawled. Performing a full scan was possible a few years ago [14], but it can be prohibitively long nowadays. New techniques must thus be developed for identifying efficiently active accounts in such settings.

1.1. Problem Formulation

Given an old snapshot, our goal is to identify a significant fraction of the items that are still alive or active now. The cost we incur is the number of fetches that are necessary to attain this a goal. A typical cost measure will be the average number of fetches per active item identified. The strategy for achieving this goal consists in producing an ordering for fetching the pages. We call *LiveRank* an ordering such that the items that are still alive tend to appear first. We consider the problem of finding an efficient LiveRank in three settings: static when it is computed solely from the snapshot and the link relations recorded at that time; sampling-based when a sampling is performed in a first phase, allowing adjustment of the ordering according to the liveness of sampled items; or finally, dynamic when it is incrementally computed as pages are fetched.

1.2. Contribution

We propose various LiveRank algorithms based on the graph structure of the snapshot. We evaluate them on two Web snapshots (from 10 to 20 millions nodes) and on a Twitter snapshot (40 million nodes). We propose several propositions based on the graph structure of the snapshot. We show that a rather simple combination of a small sampling phase and PageRank-like propagation in the remainder of the snapshot allows us to gather from 15% to 75% of the active nodes with a cost that remains within a factor of two from the optimal ideal solution.

1.3. Related Work

The process of crawling the Web has been extensively studied. A survey is given by Olston and Najork [22].

We focus here on batch crawling, for which the process starts from a given set of pages and terminates at some point.

This is classically opposed to incremental crawling for which pages are continuously fetched. In incremental crawling, one of the main tuning procedures is to balance between fetching new pages and refreshing old ones: the former increases coverage and the latter increases freshness. Both types might allow the discovery of new links to unknown new pages (old pages can change). The problem of incremental crawling has been extensively studied [9], and a thorough study on refreshing policies, and one of the first formalizations of freshness was proposed by [9]. They show the counterintuitive result that adapting the frequency of crawl proportionally to the frequency of change works poorly with respect to the overall freshness of the fetched copy from the Web. The results have been extended with

more elaborated variations of freshness. For instance, information longevity [23] considers the evolution of fragments of the content of a page.

The issue we investigate here is closer to a problem introduced by [10]: sampling is used to estimate the frequency of change per site and then to fetch a set of pages such that the overall change ratio of the set is maximized. The technique consists in estimating the frequency of page change per site and to crawl first sites with high frequency of change. This technique was improved slightly [26] by clusterizing the pages according to several features: not only their site (and other features read from the URL) but also content based features and linkage features (including PageRank and incoming degree). A change ratio per cluster is then estimated through sampling, and clusters are downloaded in descending order of the estimated values. More recently, a similar approach was investigated [25] using learning techniques and avoiding the use of sampling.

Note that these approaches mainly focus on highly dynamic pages and use various information about pages, whereas we are interested in stable nodes and we use only the graph structure, which is lighter.

With a slightly different objective, [11] investigated how to discover new pages while minimizing the average number of fetches per new page found. Their work advocates for a greedy cover heuristic when a small fraction of the new pages has to be discovered quickly. However, they recommend a heuristic based on out-degrees for gathering a large fraction of the new pages. Their framework is close to ours and inspired the cost function used in this study.

A related problem consists in estimating which pages are really valid among the "dangling" pages on the frontier of the crawled Web (those that are pointed by crawled pages but that were not crawled themselves). In [12] it is proposed to take this into account in the PageRank computation. In a similar trend, [3] propose to compute a "decay" score for each page by refining on the proportion of dead links in a page. Their goal is to identify poorly updated pages. This score could be an interesting measure for computing a LiveRank, however, its computation requires the identification of dead links. It is, thus, not clear how to both estimate it and at the same time try to avoid testing the liveness of possibly many dead pages.

Although recrawling policies have been extensively studied for web graphs, other sources of online data such as social networks are not as thoroughly covered. Yet, it is possible to similarly crawl such networks. For example, one can explore the Twitter network by fetching information about user accounts that are linked by the follower–followee relations. However, crawling is much more restricted because all the data is possessed by a single company. This makes our approach even more relevant in such contexts where gathering a large amount can take an extensively long time.

Interestingly, [20] shows, among various observations, a correlation between number of followers and PageRank. To the contrary, the activity of a user measured in number of tweets seems to be correlated more to his number of followees than his number of followers. First reported Twitter crawls include [16, 19, 20]. Recently, [13, 14] have presented preliminary studies on a complete picture of a Twitter social graph. The authors themselves claim that such extensive crawling becomes more and more difficult with time because Twitter tends to restrict its white list of IPs authorized to query its API at a high rate.

1.4. Roadmap

In the next section, we propose a simple metric to evaluate the quality of a LiveRank and we introduce several classes of possible LiveRank solutions. In Section 3, we introduce three datasets, two from the `.uk` Web and one from Twitter, and we expose how a ground truth was computed for them. Finally, in Section 4, we benchmark our LiveRanks against the datasets and discuss the results.

2. MODEL

Let $G = (V, E)$ be a graph obtained from a past crawl of a structured network. For example, G can represent

- A web graph with V representing the crawled pages and E the hyperlinks: for i, j in V, (i, j) is in E if and only if there is a hyperlink to j in i. For web graphs, edges are always directed;
- A social network with V representing the users and E social relationships between them. For social networks, edges can be undirected (symmetric relationships such as friendship) or directed (asymmetric relationship such as follower/followee).

Let n denote the size of V. At the present time, only a subset of G is still active. The meaning of *active* depends of the context and needs to be defined: alive pages for web graphs, nonidle users for social networks, etc. We call a the function that tells if nodes are active: $a(X)$ denotes the active nodes from $X \subset V$, whereas $\bar{a}(X)$ stands for $X \setminus a(X)$. Let n_a be $|a(V)|$.

The problem we need to solve is how to crawl a maximum number of pages from $a(V)$ with a minimal crawling cost. In particular, one would like to avoid crawling too many pages from $\bar{a}(V)$. If a were known, the task would be easy, but testing the activity of a node obviously requires crawling it. This is the rationale for the notion of LiveRank.

2.1. Performance Metric

Formally, any ordering can be seen as a LiveRank, so we need some performance metrics to define good LiveRanks that succeed in ranking the pages from $a(V)$ first. Following [11], we define the LiveRank cost as the average number of node retrievals necessary to obtain an active node, after a fraction $0 < \alpha \leq 1$ of the active nodes has been retrieved.

In detail, let $\mathcal{L}_i$ represent the i first pages returned by a LiveRank $\mathcal{L}$, and let $i(\mathcal{L}, \alpha)$ be the smallest integer such that $\frac{|a(\mathcal{L}_i)|}{n_a} \geq \alpha$. The cost function of $\mathcal{L}$, which depends on α, is then defined by:

$$\text{cost}(\mathcal{L}, \alpha) = \frac{i(\mathcal{L}, \alpha)}{\alpha n_a}.$$

A few remarks on the cost function:

- It is always at greater than or equal to 1. An ideal LiveRank would perfectly separate $a(V)$ from the rest of the nodes, so its cost function would be 1. Without precognition, this requires testing all pages, which is exactly what we would like to avoid. The cost function allows us to capture this dilemma.

- Keeping a low cost becomes difficult as α gets close to 1: without some clairvoyant knowledge, capturing *almost* all active nodes is almost as difficult as capturing all actives nodes. For that reason, one expects that when α gets close to 1, the set of nodes any real LiveRank will need to crawl will tend to V, leading to an asymptotical cost $\frac{n}{n_a}$. This will be verified in Section 4.
- Finally, one might have noticed that the cost function uses $n_a = |a(V)|$, for which an exact value requires a full knowledge of active nodes. This is not an issue here because we will perform our evaluation on datasets in which a is known. For use on datasets without ground truth, one could either use an estimation of n_a based on a sampling or use a nonnormalized cost function (for instance the fraction of active nodes obtained after i retrievals).

2.2. PageRank

Many of the LiveRanks proposed here are based on some variants of PageRank. PageRank is a link analysis algorithm introduced in [24] and used by the Google Internet search engine. It assigns a numerical importance to each page of a web graph. It uses the structural information from G to attribute importance according to the following (informal) recursive definition: *a page is important if it is referenced by important pages*. Concretely, to compute PageRank value, denoted by the row vector Y, one needs to find the solution of the following equation:

$$Y = dYA + (1 - d)X, \tag{2.1}$$

where A is a substochastic matrix derived from the adjacency matrix of G, $d < 1$ is a so-called damping factor (often set empirically to $d = 0.85$), and $X \gneqq 0$ is a *zap vector*; X represents a kind of importance *by default* that is propagated from nodes to nodes according to A with a damping d.

Computation of PageRank vectors has being widely studied. Several specific solutions were proposed and analyzed [21, 4] including the power method [24], adaptation [17], extrapolation [15, 18], the adaptive online method [2], etc.

We now present the different LiveRanks that we will consider in this article. We broadly classify them in to three classes: static, sample-based, and dynamic.

2.3. Static LiveRanks

Static LiveRanks are computed offline using uniquely the information from G. That makes them very basic, but also very easy to use in a distributed way: given p crawlers of similar capacities, if $\mathcal{L} = (l_1, \ldots, l_n)$, simply assign the task of testing node l_i to crawler i mod p.

We propose the following three static LiveRanks:

Random permutation. (R) will serve both as a reference and as a building block for more advanced LiveRanks. R ignores any information from G, so its cost should be in average $\frac{n}{n_a}$, with a variance that tends to 0 as α tends to 1. We expect good LiveRanks to have a cost function significantly lower than cost(R).

Decreasing Indegree ordering. (I) is a simple LiveRank that we expect to behave better than a random permutation. Intuitively, a high indegree can mean some importance, and important pages may be more robust. Also, older nodes should have more incoming

edges (in terms of correlation), so high-degree nodes can correspond to nodes that were already old at the time G was crawled, and old nodes might last longer than younger ones. Sorting by degree is the easiest way to exploit these correlations.

PageRank ordering. (P) pushes forward the *indegree* idea. The intuition is that nodes that are still active are likely to point toward nodes that are also still active, even considering only old edges. This suggests using a PageRank-like importance ranking. In absence of further knowledge, we propose to use the solution of (2.1) using $d = .85$ (typical value for web graphs) and X uniform on V.

Note that it is very subjective to evaluate PageRank as an importance ranking, because importance should be ultimately validated by humans. However, the quality of PageRank as a static LiveRank is straightforward to verify, for instance, using our cost metric.

The possible existence of correlation between Indegree (or PageRank) and activity will be investigated in Section 3.3.

2.4. Sample-Based LiveRanks

Using a LiveRank consists in crawling V in the prescribed order. During the crawl, the activity function a becomes partly available, and the obtained information could be used to enhance the retrieval. Following that idea, we consider here a two-step, sample-based approach: we first fix a testing threshold z and test z items following a static LiveRank (such as R, I, or P). For the set Z of nodes tested, called *sample set* or *training set*, $a(Z)$ is known, which allows us to recompute the LiveRank of the remaining untested nodes.

Because the sampling uses a static LiveRank, and the adjusted new LiveRank is static as well, sample-based LiveRanks are still easy to use in a distributed way because the crawlers need to receive crawl instructions on only two occasions.

Notice that in the case of the sampling LiveRank being a random permutation, $|a(Z)|\frac{n}{z}$ can be used as an estimate for n_a. This can, for instance, be used to decide when to stop crawling if we desire to identify αn_a active nodes in $a(V)$.

2.4.1. Simple Adaptive LiveRank (P_a). When a node is active, we can assume it increases the chance that nodes it points to in G are also active, and that activity is transmitted somehow through hyperlinks. Following this idea, a possible adaptive LiveRank consists in taking for X in (2.1) the uniform distribution on $a(Z)$. This diffusion from such an initial set can be seen as a kind of breadth-first traversal starting from $a(Z)$, but with a PageRank flavor.

2.4.2. Double Adaptive LiveRank ($P_a^{+/-}$). The simple adaptive LiveRank does not use the information given by $\bar{a}(Z)$. One way to do this is to calculate an "anti"-PageRank based on $\bar{a}(Z)$ instead of $a(Z)$. This ranking would represent a kind of diffusion of idleness, the underlying hypothesis being that idle nodes point to nodes that tend to be idle, too. As a result, we obtain a new LiveRank by combining these two PageRanks. After having tested several possible combinations not discussed in this article, we empirically chose to weight each node by the ratio of the two sample-based PageRanks, after having set all null entries of the anti-PageRank equal to the minimal nonnull entry.

2.4.3. Active-site First LiveRank (ASF). To compare with previous work, we propose the following variant inspired by the strategy for finding pages that have changed in a recrawl [11]. Their algorithm is based on sampling for estimating page change rate for each website and then crawling sites by decreasing change rate. In details, active-site first (ASF) consists in partitioning Z into websites determined by inspecting the URLs. We thus obtain a collection $Z_1, \ldots, Z_p$ of sets. For each set Z_i corresponding

to some site i, we obtain an estimation $|a(Z_i)|/|Z_i|$ of its activity (i.e., the fraction of active pages in the site). We then sort the remaining URLs by decreasing site activity. Of course, this technique works only for web graphs and can hardly be adjusted to other networks.

2.5. Dynamic LiveRanks

Instead of using the acquired information just one time after the sampling, Dynamic LiveRanks are continuously computed and updated on the fly along the entire crawling process. On the one hand, this gives them real-time knowledge of a, but on the other hand, as the dynamic LiveRank might evolve all the time, they can create synchronization issues when used by distributed crawlers.

As for sample-based LiveRanks, dynamic LiveRanks use a training set Z of z nodes from a static LiveRank. This allows bootstraping the adjustment by giving a nonempty knowledge of a and prevents the LiveRank from focusing on only a small subset of V.

Breadth-first search (BFS) With BFS, we aim at taking direct advantage of the possible propagation of activity. The BFS queue is initialized with the (uncrawled) training set Z. The next node to be crawled is popped from the queue following first-in-first-out (FIFO) rule. If the selected node appears to be active, all of its uncrawled outgoing neighbors are pushed to the end of the queue. When the queue is empty, we pick the unvisited node with highest PageRank.[1]

Active indegree (AI) BFS uses a simple FIFO queuing to determine the processing order. We now propose AI, which provides a more advanced node-selection scheme. For AI, each node in the graph is associated with an *activity score* value indicating how many reported active nodes point to it. These values are set to zeros at the beginning and are always kept up-to-date. AI is initialized by testing Z: each node in $a(Z)$ will increment the associated values of its outgoing neighbors by one. After Z is tested, the next node to be crawled is simply the one with the highest activity score (in case of equality, to keep things consistent, we pick the node with the highest PageRank). Whenever a new active node is found, we update the activity scores of its untested neighbors.

With Dynamic LiveRank, it is natural to think of a last variant, a dynamic PageRank-based strategy in which the PageRank vector is recursively computed. Starting from a uniform distribution on $a(Z)$, we obtain X in (2.1). Then a new teleportation vector is constructed as a uniform distribution on largest value entries of X, i.e., those which are considered probably active after the first diffusion of $a(Z)$. The process continues and X is updated iteratively. However, after some experimentations, we realized that this method is not efficient because it cannot escape from the locality of $a(Z)$.

3. DATASETS

We chose to evaluate the proposed LiveRanks on existing datasets of the Web and Twitter available on the webgraph platform [8, 6, 5]. In this section, we present the datasets, describe how we obtained the activity function a, and observe the correlations between a, indegree, and PageRank.

[1]We tested several other natural options and observed no significant impact.

Status	Description	Number of Pages	Percentage
Code HTTP 404	Page not found	6, 467, 219	34.92%
No answer	Host not found	4, 470, 845	24.14%
Code HTTP 301	Redirection	3, 455, 923	18.66%
Target 301	Target of redirection	20, 414	0.11%
Code HTTP 200	Page exists	2, 365, 201	12.77%
True 200	**Page really exists**	1, 164, 998	**6.29%**
Others (403,. . .)	Other error	1, 761, 298	9.51%
Total	Graph size	18, 520, 486	100%

Table I Status of webpages in `uk-2002`, crawled in December 2013.

3.1. Web Graph Datasets

For the study of web graphs, we focused on snapshots of the British domain `.uk`.

3.1.1. uk-2002 Dataset. The main dataset we use is the web graph `uk-2002`[2] from UbiCrawler [5]. This 2002 snapshot contains 18,520,486 pages and 298,113,762 hyperlinks.

The preliminary task is to determine a. For web graphs, it describes the liveness of the pages of the snapshot. For each URL, we have performed a GET request and hopefully obtained a corresponding HTTP code. Our main findings are these:

- One third of the total pages are no longer available today, the server returns error `404`.
- One fourth have a DNS problem (which probably means the website is also dead).
- For one fifth of the cases, the server sends back the redirection message `301`. Most redirections for pages of an old site lead to the root of a new site. If we look at the proportion of distinct pages alive at the end of redirections, it is as low as 0.1%.
- Less than 13% of pages return the code `200` (success). However, we found out that half of them actually display some text mentioning that the page was not found. To handle this issue, we have fully crawled all the pages of the dataset with code `200` and filtered out pages whose title or content have either `Page Not Found` or `Error 404`.

The results are summarized in Table I. In the end, our methodology led to finding 1,164,998 alive pages, accounting for 6.4% of the dataset.

3.1.2. uk-2006 Dataset. The settings of `uk-2002` are rather adversarial (old snapshot with relatively few alive pages), so we wanted to evaluate the impact of LiveRanks on shorter time scales. In absence of fresh-enough available datasets, we used the DELIS dataset [7], a series of twelve continuous snapshots[3] starting from 06/2006 to 05/2007 (one-month intervals). We set G to the first snapshot (06/2006). It contains 31,316,403 nodes and 813,807,972 hyperlinks. We then considered the last snapshot (05/2007) as "present time," setting the active set $a(V)$ as the intersection between the two snapshots. With this methodology, we hope to have a good approximation of a after a one-year period. For this dataset, we obtained $n_a = 11, 142, 177$ "alive" nodes, representing 35.56% of the graph.

[2]http://law.di.unimi.it/webdata/uk-2002/

[3]http://law.di.unimi.it/webdata/uk-union-2006-06-2007-05/

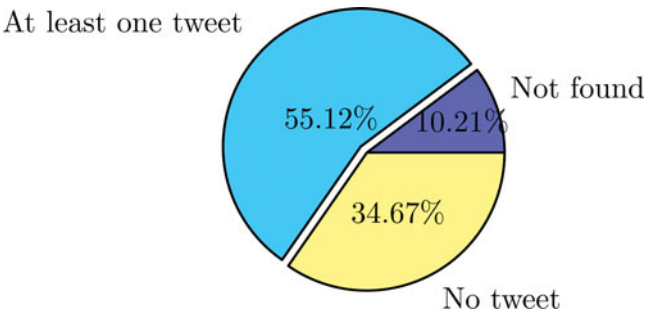


Figure 1 Statistics of the `twitter-2010` dataset.

3.2. Twitter Dataset

Finally, we used the dataset `twitter-2010`, first introduced in [20].[4] The graph contains roughly 42 million Twitter user accounts and 1.5 billion follower–followee relationships among them. Arcs in the graph are directed from followers to followees: there is an arc from node x to y if user x follows y. This orientation convention is in line with a PageRank approach: a user is important when followed by important users. Note that information (tweets) traverses arcs in the opposite direction, from followees to followers.

We consider that a user is active if he/she has posted a tweet recently. For that purpose, we can query the Twitter interface to recover the timestamp of the last tweet of the user associated with a given identifier. Recovering the timestamps of all 41 millions users using Twitter API [1] would be extremely slow: when we made our measurements (05/2014), an authorized Twitter account was limited to 350 API requests/hour so querying all the accounts would have taken 13 years. Although this is one of the main reasons for designing a good LiveRank, we still need a full crawl to build a ground truth. To overcome this obstacle, we worked around the API limitation by using a browser-like crawler to recover each user timeline as if a regular browser were connecting to Twitter front servers. This is possible because the timestamp of the last entry can easily be scrapped from the HTML structure of the returned documents. However, such an approach becomes much more difficult for complex queries and might also be detected and prevented by Twitter in the future.

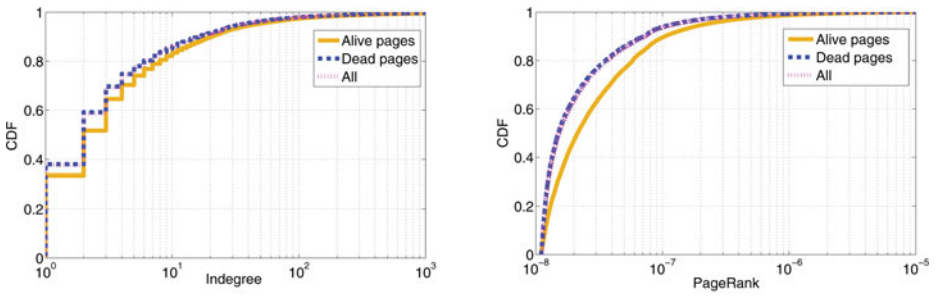
Having tested all nodes, we found three main categories of users corresponding to those who (i) no longer exist, (ii) have no tweet at all, and (iii) have tweeted at least once before the crawling time. Figure 1 shows the relative proportion of each category.

For users with at least one tweet, we extracted the timestamp of their last tweet. After considering the cumulative distribution of last-tweet timestamps, we arbitrarily decided to set the activity threshold to six months: a user is active if he/she has tweeted during the last six months. With these settings, we obtained a list of 7,300,399 (17.53%) active users, serving as ground truth for LiveRank evaluation.

3.3. Correlations

The rationale behind the LiveRanks (I) and (P) is the assumption that the activity of nodes is correlated to the graph structure of the snapshot, so that a node with high indegree or PageRank has more chances to stay active.

[4] http://law.di.unimi.it/webdata/twitter-2010/



(a) Indegree CDF (b) PageRank CDF

Figure 2 Impact of liveness to Indegree/PageRank distribution for `uk-2002`.

To validate this, we plot in Figure 2 the cumulative distribution of indegree (Figure 2a) and PageRank (Figure 2b) for alive, dead, and all pages of the `uk-2002` dataset. We observe that the curve for active nodes is slightly shifted to the right compared to the other curves in the figure active users tend to have slightly higher indegree and PageRank than the overall population. The bias is larger for PageRank, suggesting that LiveRank (P) should perform better than LiveRank (I) for web graphs.

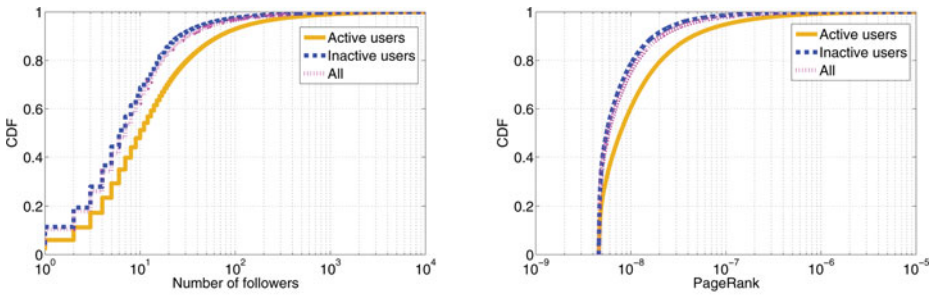
Figure 3 presents the same results for the `twitter-2010` dataset. Although the curves are qualitatively similar, the bias comparison is not as clearly in favor of PageRank.

We will now measure how this bias impacts the cost function of corresponding LiveRanks.

4. LIVERANKS EVALUATION

After having proposed several LiveRanks in Section 2 and described our datasets in the previous section, we can now benchmark our proposals.

All our evaluations are based on representations of the cost functions. In each plot, the x-axis indicates the fraction α of active nodes we aim to discover, and the y-axis corresponds to the relative cost of the crawl required to achieve that goal. A low curve indicates an efficient LiveRank. As said previously, an ideal LiveRank would achieve a constant cost of 1; a random LiveRank is quickly constant with an average cost n/n_a; any nonclairvoyant LiveRank will tend to cost n/n_a as α goes to 1.



(a) Followers CDF (b) PageRank CDF

Figure 3 Impact of activity to Followers/PageRank for `twitter-2010`.

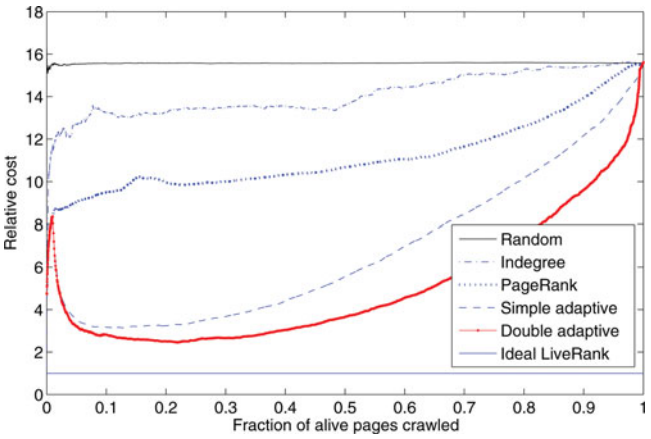


Figure 4 Main results on static and sample-based LiveRanks (uk-2002).

4.1. Evaluation on Web Graphs

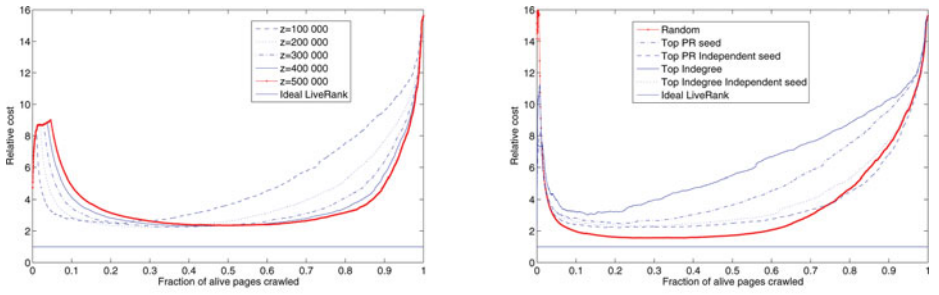
We focus here mainly on the uk-2002 dataset. Unless otherwise specified, the training set contains the $z = 100,000$ pages of higher (static) PageRank.

4.1.1. Static and Sample-based LiveRanks. We first evaluate the results of static and sample-based LiveRanks. The results are displayed in Figure 4. For static LiveRanks, we see as expected that a random ordering gives an almost constant cost equal to $\frac{n}{n_a} \approx 15.6$. Indegree ordering (I) and PageRank (P) significantly outperform this result, PageRank being the best of the three: it is twice more efficient than random for small α, and still performs approximately 30% better when up to $\alpha = 0.6$. We then notice that we can get much better costs with sample-based approaches, the double-adaptive LiveRank $P_a^{+/-}$ giving a significant improvement over the simple-adaptive one, P_a. The use of $P_a^{+/-}$ allows improving the ordering by a factor of 6 approximately around $\alpha = 0.2$ with a cost of 2.5 fetches per active node found. The cost for gathering half of the alive pages is less than 4, and for 90% it stays under 10.

4.1.2. Quantitative and Qualitative Impact of the Training Set. We study in Figure 5 the impact of the training sets on sample-based LiveRanks. Results are shown for $P_a^{+/-}$ but similar results were obtained for P_a.

Figure 5a shows the impact of the size z of the sampling set (sampling the top PageRank pages). We observe some trade-off: as the sampling set grows larger, the initial cost increases because the sample does not use any fresh information, but it results in a significant increment of efficiency in the long run. For this dataset, taking a large training set ($z = 500,000$) allows reducing the cost of the crawl for $\alpha \geq 0.4$, and maintains a cost less than 4 for up to 90%.

Another key aspect of the sampling phase is the qualitative choice of the sample set. Using $z = 100,000$, we can observe in Figure 5b that the performance of double adaptive $P_a^{+/-}$ is further improved by using a random sample set rather than selecting it according to the PageRank or by decreasing indegree. We believe that the reason is that a random sample avoids a locality effect in the sampling set because high PageRank pages tend to concentrate in some local parts of the graph. To verify that, we tried to modify Indegree and PageRank



(a) Impact of the size z (b) Impact of the selection of Z

Figure 5 Impact of the training set (uk-2002).

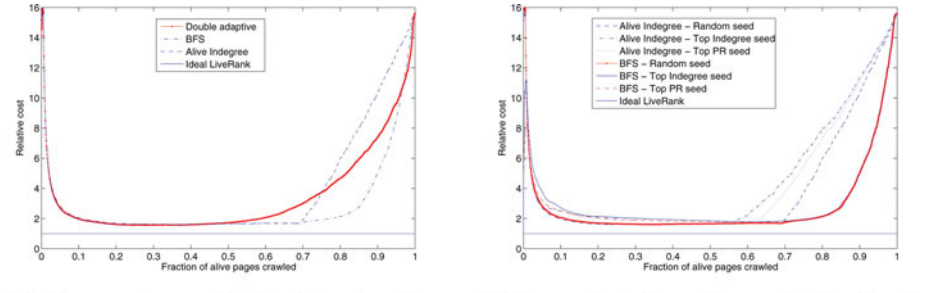
selection to avoid the selection of neighbor pages. The results (not displayed here) show a significant improvement while staying less efficient than using a random sample.

To summarize, double-adaptive LiveRank through random sampling seems to offer a very low cost, within a factor of 2 from optimal for a large range of values α.

4.1.3. Dynamic LiveRanks. We then consider the performance of fully dynamic strategies, using the double-adaptive LiveRank with random training set as a benchmark. The results are displayed in Figure 6a. We see that BFS and AI perform similarly to double adaptive $P_a^{+/-}$ for low α and can outperform it for large α (especially BFS). BFS begins to significantly outperform double-adaptive LiveRank for $\alpha \geq 0.5$. However, if one needs to gather half of the active pages or fewer, double-adaptive is still the best candidate because it is much simpler to operate, especially with a distributed crawler.

Additionally, Figure 6b shows the impact of different sampling sets on BFS and AI. Except for high values of α where a random sampling outperforms other strategies, the type of sampling does not seem to affect the two dynamic LiveRanks as much as was observed for the double-adaptive LiveRank.

4.1.4. uk-2006 Dataset. We have repeated the same experiments on the dataset uk-2006, in which the update interval is only one year. Figure 7 shows the results for static and sample-based LiveRanks, using $z = 200,000$ (because the dataset is larger) and random sampling. The observations are qualitatively quite similar to uk-2002. The main difference is that all costs are lower due to a higher proportion of alive pages ($\frac{n}{n_a} \approx 2.81$). The



(a) Comparison with double-adaptive (b) Impact of Z on dynamic LiveRanks

Figure 6 Performance of dynamic LiveRanks (uk-2002).

double-adaptive version still gives the lower relative cost among static and sample-based LiveRanks, staying under 1.4 for a wide range of α.

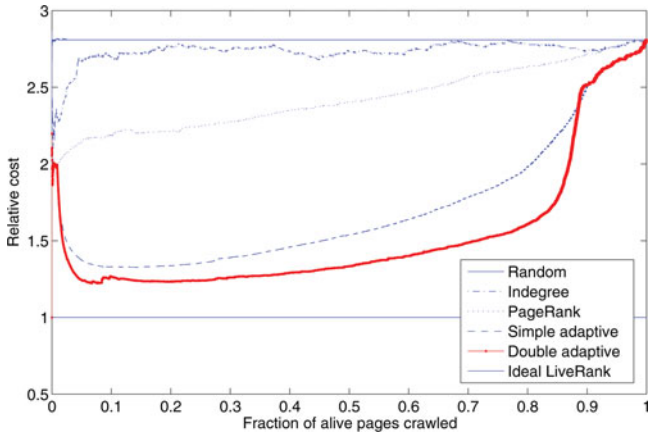


Figure 7 uk-2006 main evaluation results.

4.1.5. Comparison with a Site-based Approach. To benchmark with techniques from previous work for finding webpages that have been updated after a crawl, Figure 8 compares double-adaptive $P_a^{+/-}$ to ASF with random sampling. The number of random pages tested in each site and the overall number of tests are the same for both methods. Note that given the budget z, it was not possible to sample small websites. Unsampled websites are crawled after the sampled ones.

We see that for α greater than 0.9, ASF performs like a random LiveRank. This corresponds to the point at which all sampled websites have been crawled. That effect aside, the performance of ASF is not as good as that of double-adaptive LiveRank for earlier α. In the end, ASF beats $P_a^{+/-}$ for only a small range of α, between 0.7 and 0.85, and the gain within that range stays limited.

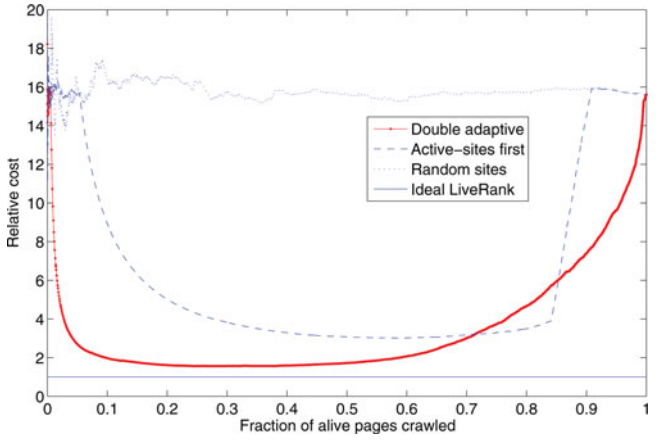


Figure 8 Comparison with active-site first LiveRank (uk-2002).

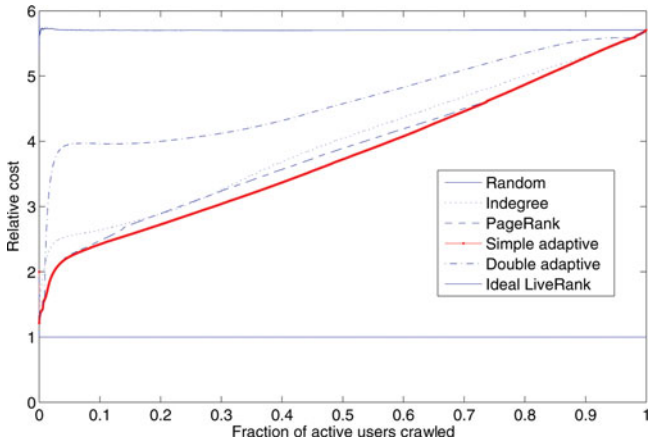


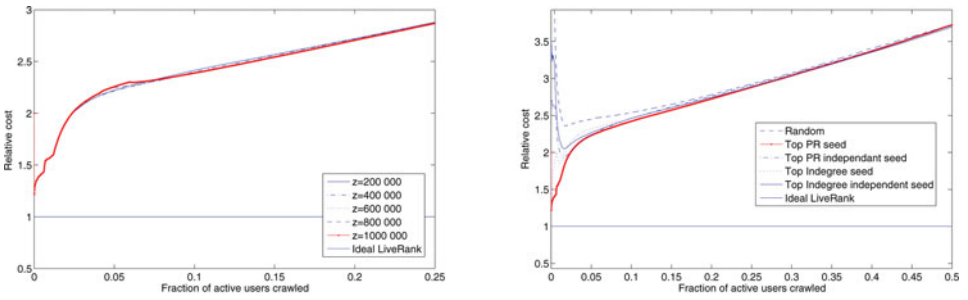
Figure 9 Main results on static and sample-based LiveRanks (`twitter-2010`).

4.2. Evaluation on Twitter

As discussed before, the Twitter graph has structural properties distinct from web graphs. In this part we analyze how these differences affect the performance of LiveRanks.

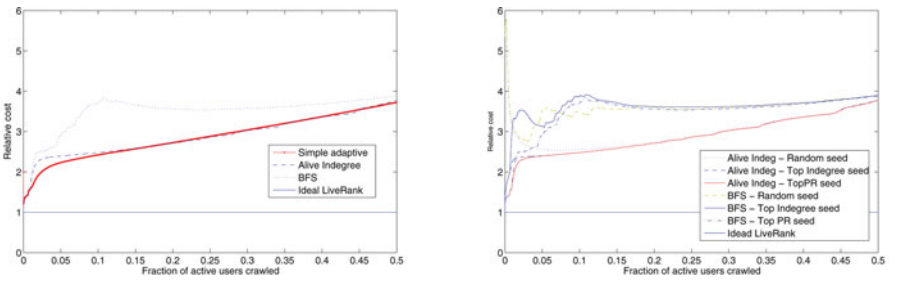
4.2.1. Static and Sampled-based LiveRanks. Figure 9 compares the static and sample-based LiveRanks. A first observation is that the double-adaptive LiveRank $P_a^{+/-}$ performs very poorly compared to the other LiveRanks, including Indegree I. It indicates that if the intuition of some death propagation was relevant for web graphs (it was a convenient way to spot dead websites for instance), this is not the case for Twitter: the fact that followers become inactive does not seem to have an impact on the activity of the followees. In the end, the simple adaptive LiveRank P_a has the best performance, closely followed by the static LiveRanks (P) and (I). Each of the three of them has a cost function that seems to grow roughly linearly between 2 and 4 as α goes from 0 to 0.6.

4.2.2. Quantitative and Qualitative Impact of the Training Set. In Figure 10a, we vary the size of the training set, ranging from $z = 200,000$ to $z =$1,000,000. Results indicate that the cost function is almost not affected by z as long as it is high enough. Compared to the results observed on web graphs, this means that taking a large training set



(a) Impact of the size z (b) Impact of the selection of Z

Figure 10 Impact of the training set (`twitter-2010`).



(a) Comparison with double adaptive (b) Impact of Z on dynamic LiveRanks

Figure 11 Performance of dynamic LiveRanks (`twitter-2010`).

(i) will not burden the cost function for small α. This likely comes from the fact that the sampling set is PageRank-based by default, and the static PageRank is already close to the best LiveRank we can get;
(ii) will not improve the performance for large α either, meaning that no significantly useful knowledge is obtained after some threshold. This relative independence with respect to z is another qualitative difference compared to web graphs.

Figure 10b shows the impact of training set types on simple-adaptive LiveRank P_a. Unlike web graphs, where random sampling dominates others, in social networks the training set filled by PageRank is the best, whereas the random seed is worse. This can be interpreted as a result of a weaker structural locality (i.e., no highly correlated clusters such as websites for web graphs), so that activeness is more concentrated around important Twitter individual users who should be considered as soon as possible.

4.2.3. Dynamic LiveRanks. In Figure 11a, we compare the simple-adaptive PageRank P_a with the dynamic LiveRanks. All of them are initialized with default values (PageRank sampling of size $z = 100,000$). P_a stays the best option: it is slightly better than AI and much more efficient than BFS. However, for web graphs, dynamic LiveRanks could still be preferred for some settings, it seems that in the context of Twitter it is never the case, especially considering their deployment complexity in a distributed crawler.

Finally, Figure 11b indicates the impact of different training sets on the two dynamic LiveRanks. It confirms that the combination of AI and a PageRank-ordered training set gives the best results for that type of LiveRanks, which is still not enough to compete against P_a.

5. CONCLUSION

In this article, we investigated how to efficiently retrieve large portions of alive pages from an old crawl using orderings we called LiveRanks. We observed that PageRank is a good static LiveRank, which can be significantly improved by first testing a small fraction of the pages for adjustment in a sample-based approach.

Compared to previous work on identifying modified pages, our technique performs similarly for a given large desired fraction (around 80%) when compared to the LiveRank algorithm inspired by the technique in [10]. However, outside that range, our method outperforms this technique. Interesting future work could reside in using our techniques

for the problem exposed in [10] (identification of pages that have changed) and comparing them with the website sampling approach.

Another advantage of our technique is the possibility that it can be applied to more general types of structured networks, such as Twitter. However, it seems that the choice of an appropriate LiveRank is closely related to the type of network.

Interestingly, we could not get significant gain when using fully dynamic LiveRanks. As noticed before, each of the two phases of the sample-based approach can be easily parallelized through multiple crawlers, whereas this would be much more difficult with a fully dynamic approach. The sample-based method could, for example, be implemented with in two rounds of a simple map-reducing program, whereas the dynamic approach requires continuous exchanges of messages between the crawlers.

Our work establishes the possibility of efficiently recovering a significant portion of the active nodes of an old snapshot and advocates for the use of an adaptive sample-based PageRank for obtaining an efficient LiveRank.

To conclude, we emphasize again that the LiveRank approach proposed in this article is very generic, and its field of applications is not limited to web graphs or Twitter. It can be straightforwardly adapted to any online data with similar linkage that enables crawling.

FUNDING

The work presented in this article was carried out at LINCS (http://www.lincs.fr) and was funded by the European project Reveal (http://revealproject.eu/).

REFERENCES

[1] Twitter graph 2010. https://dev.twitter.com/.

[2] S. Abiteboul, M. Preda, and G. Cobena. “Adaptive On-line Page Importance Computation.” In *Proceedings of the 12th International Conference on World Wide Web*, WWW ’03, pp. 280–290. New York, NY: Association for Computing Machinery Press, 2003.

[3] Z. Bar-Yossef, A. Z. Broder, R. Kumar, and A. Tomkins. “Sic transit gloria telae: Towards an Understanding of the Web’s Decay.” In *Proceedings of the 13th International Conference on World Wide Web,* WWW ’04, pp. 328–337. New York, NY: Association for Computing Machinery Press, 2004.

[4] M. Bianchini, M. Gori, and F. Scarselli. “Inside Pagerank.” *ACM Trans. Internet Technol.* 5:1 (2005), 92–128.

[5] P. Boldi, B. Codenotti, M. Santini, and S. Vigna. “Ubicrawler: A scalable Fully Distributed Web crawler.” *Software: Practice & Experience* 34:8 (2004), 711–726.

[6] P. Boldi, M. Rosa, M. Santini, and S. Vigna. Layered Label Propagation: A Multiresolution Coordinate-Free Ordering for Compressing Social Networks. In *Proceedings of the 20th International Conference on World Wide Web*. New York, NY: Association for Computing Machinery Press, 2011.

[7] P. Boldi, M. Santini, and S. Vigna. “A Large Time-Aware Graph.” *SIGIR Forum* 42:2 (2008), 33–38.

[8] P. Boldi and S. Vigna. “The WebGraph Framework I: Compression Techniques.” In *Proc. of the Thirteenth International World Wide Web Conference (WWW 2004)*, pp. 595–601. New York, NY: Association for Computing Machinery Press, 2004.

[9] J. Cho and H. Garcia-Molina. “Effective Page Refresh Policies for Web Crawlers.” *ACM Transactions on Database Systems (TODS)* 28:4 (2003), 390–426.

[10] J. Cho and A. Ntoulas. "Effective Change Detection Using Sampling." In *Proceedings of the 28th International Conference on Very Large Data Bases*, pp. 514–525. New York, NY: Association for Computing Machinery, 2002.

[11] A. Dasgupta, A. Ghosh, R. Kumar, C. Olston, S. Pandey, and A. Tomkins. "The Discoverability of the Web." In *Proceedings of the 16th International Conference on World Wide Web*, pp. 421–430. New York, NY: Association for Computing Machinery, 2007.

[12] N. Eiron, K. S. McCurley, and J. A. Tomlin. "Ranking the Web Frontier." In *Proceedings of the 13th International Conference on World Wide Web*, pp. 309–318. New York, NY: ACM, 2004.

[13] M. Gabielkov and A. Legout. "The Complete Picture of the Twitter Social Graph." In *Proceedings of the 2012 ACM Conference on CoNEXT Student Workshop*, pp. 19–20. New York, NY: Association for Computing Machinery Press, 2012.

[14] M. Gabielkov, A. Rao, and A. Legout. "Studying Social Networks at Scale: Macroscopic Anatomy of the Twitter Social Graph." Paper presented at ACM Sigmetrics 2014, Austin, TX, USA, June 2014.

[15] T. Haveliwala, A. Kamvar, D. Klein, C. Manning, and G. Golub. "Computing pagerank using power extrapolation." Technical Report, Stanford University, 2003.

[16] A. Java, X. Song, T. Finin, and B. Tseng. "Why we Twitter: Understanding Microblogging Usage and Communities." In *Proceedings of the 9th WebKDD and 1st SNA-KDD 2007 Workshop on Web Mining and Social Network Analysis*, pp. 56–65. New York, NY: Association for Computing Machinery Press, 2007.

[17] S. Kamvar, T. Haveliwala, and G. Golub. "Adaptive Methods for the Computation of Pagerank." Technical Report 2003-26, Stanford InfoLab, April 2003.

[18] S. D. Kamvar, T. H. Haveliwala, C. D. Manning, and G. H. Golub. "Extrapolation Methods for Accelerating Pagerank Computations." In *Proceedings of the 12th International Conference on World Wide Web*, WWW '03, pp. 261–270, New York, NY: Association for Computing Machinery Press, 2003.

[19] B. Krishnamurthy, P. Gill, and M. Arlitt. "A Few Chirps About Twitter." In *Proceedings of the First Workshop on Online Social Networks*, pp. 19–24. New York, NY: Association for Computing Machinery Press, 2008.

[20] H. Kwak, C. Lee, H. Park, and S. Moon. "What is Twitter, a Social Network or a News Media?" In *Proceedings of the 19th International Conference on World Wide Web*, pp. 591–600. New York, NY: Association for Computing Machinery Press, 2010.

[21] A. N. Langville and C. D. Meyer. "Deeper Inside Pagerank." *Internet Mathematics* 1:3 (2004), 335–380.

[22] C. Olston and M. Najork. "Web Crawling." *Foundations and Trends in Information Retrieval*, 4:3 (2010), 175–246.

[23] C. Olston and S. Pandey. "Recrawl Scheduling Based on Information Longevity." In *Proceedings of the 17th International Conference on World Wide Web*, pp. 437–446. New York, NY: Association for Computing Machinery Press, 2008.

[24] L. Page, S. Brin, R. Motwani, and T. Winograd. In "The PageRank citation Ranking: Bringing Order to the Web" Technical Report number 1999-66, Stanford InfoLab, November 1999.

[25] K. Radinsky and P. N. Bennett. "Predicting Content Change on the Web." In *Proceedings of the Sixth ACM International Conference on Web Search and Data Mining*, pp. 415–424. New York, NY: Association for Computing Machinery Press, 2013.

[26] Q. Tan, Z. Zhuang, P. Mitra, and C. L. Giles. "A Clustering-Based Sampling Approach for Refreshing Search Engine's Database." In *Proceedings of the 10th International Workshop on the Web and Databases*, 2007.