Fast algorithms for single-node and personalized PageRank

Krishna N Agaram
RnD project with Prof. Akash Kumar

Sep-Nov 2023

Abstract

This is a survey of fast algorithms for single-node and personalized PageRank, mainly focusing on the work of Wang et al. [WW23]. We introduce PageRank from a probabilistic perspective, mainly due to its simplicity and the fact that it makes many useful intermediate PageRank objects seem natural. The papers of Andersen et al. [ACL06] and Lofgren et al. [LG14] are then surveyed, appreciating similarities and dissimilarities. We then survey in detail the work of Wang et al. [WW23], which will be the main focus of this report.

1 Preliminaries

Consider a directed graph $G = (V, E)$ with $|V| = n$ and $|E| = m$. We denote its adjacency by A and the diagonal matrix of the vertices' degrees by D. The random walk matrix M is then $M = D^{-1}A$. Thus,

$$M(u, v) = \frac{\mathbb{1}_{[(u,v)\in E]}}{d_u}.$$

All probability vectors will be *row* vectors as in the standard Markov chain literature. We also treat functions $f : V \to \mathbb{R}$ as row vectors when convenient. We use the notation $\mathbf{1}_v$ to denote a **column** vector that has a 1 in the v^{th} entry and is 0 everywhere else.

Consider random variables $X_0, X_1, \ldots$ on V and $S_0, S_1, \ldots$ on $\{0, 1\}$ with the following distribution:

$$X_0 \sim \mu_0$$

$$\mathbb{P}\left[X_{t+1}|X_t, \neg S_t\right] = \mathbb{1}_{[(X_t,X_{t+1})\in E]}\frac{1}{d_{X_t}}$$
$$\mathbb{P}\left[X_{t+1}|X_t, S_t\right] = \mathbb{1}_{[X_t=X_{t+1}]}$$

$$\mathbb{P}\left[S_{t+1}|\neg S_t, \star\right] = \alpha \;\forall \star \subseteq \{X_0, \ldots\}$$
$$\mathbb{P}\left[S_{t+1}|S_t, \star\right] = 1 \;\forall \star \subseteq \{X_0, \ldots\}$$

Essentially, the random variables X_t trace a random walk on G, and each variable S_t decides whether to stop the random walk at time t or not. Note that this is different from a lazy random walk, thanks to the last two equations above. The parameter $\alpha \in (0, 1)$ is called the *teleportation probability*. This random walk is called an α-random walk on G. Given $u, v \in V$ and $\ell \in \mathbb{N}$, consider random variable $u \overset{\ell}{\rightsquigarrow} v$ defined by

$$u \overset{\ell}{\rightsquigarrow} v = \begin{cases} 1 & \text{if } X_0 = u, X_\ell = v, S_0 = \cdots = S_{\ell-1} = 0 \\ 0 & \text{otherwise} \end{cases}$$

We denote

$$\pi_u^{(\ell)}(v) = \pi^{(\ell)}(u, v) = \mathbb{P}\left[u \overset{\ell}{\rightsquigarrow} v, S_\ell\right] = \alpha\mathbb{P}\left[u \overset{\ell}{\rightsquigarrow} v\right]. \tag{1}$$

Lemma 1 (one-step recurrence relation).

$$\pi^{(\ell+1)}(u,v) = (1-\alpha)\sum_{w\in V}\pi^{(\ell)}(u,w)M(w,v) = (1-\alpha)\sum_{w\in \mathrm{IN}(v)}\frac{1}{d_w}\pi^{(\ell)}(u,w). \tag{2}$$

Proof. The left hand side is

$$\begin{aligned}\pi^{(\ell+1)}(u,v) &= \mathbb{P}\left[\exists w.\, u \overset{\ell}{\rightsquigarrow} w, S_\ell = 0, X_{\ell+1} = v, S_{\ell+1}\right] \\ &= \sum_{w\in V}\mathbb{P}\left[S_{l+1}\middle|X_{\ell+1}=v, S_\ell = 0, u\overset{\ell}{\rightsquigarrow} w\right]\mathbb{P}\left[X_{\ell+1}=v\middle|S_\ell=0, u\overset{\ell}{\rightsquigarrow} w\right]\mathbb{P}\left[S_\ell=0\middle|u\overset{\ell}{\rightsquigarrow} w\right]\mathbb{P}\left[u\overset{\ell}{\rightsquigarrow} w\right] \\ &= (1-\alpha)\sum_{w\in V}\pi^{(\ell)}(u,w)M(w,v).\end{aligned}$$

□

Defining the matrix $\pi^{(\ell)}$ by $\pi^{(\ell)}(u,v) = \pi^{(\ell)}(u,v)$, the lemma above can be written as $\pi^{(\ell+1)} = (1-\alpha)\pi^{(\ell)}M$. Noting that $\pi^{(0)} = \alpha I$, we have

$$\pi^{(\ell)} = \alpha(1-\alpha)^\ell M^\ell. \tag{3}$$

The row vector $\pi^{(\ell)}(u,\cdot)$, called the *personalized ℓ-hop PageRank vector* of u, is easily found from $\pi^{(\ell)}$ as $\pi^{(\ell)}(u,\cdot) = \mathbf{1}_u^T\pi^{(\ell)}$. It represents the probability of an ℓ-length α-random walk starting at u ending at each vertex. The *personalized PageRank vector* π_u is defined by

$$\pi_u(v) = \sum_{\ell=0}^{\infty}\pi_u^{(\ell)}(v) = \mathbf{1}_u^T M_\alpha, \tag{4}$$

where

$$M_\alpha = \sum_{l=0}^{\infty}\pi^{(\ell)} = \alpha\sum_{\ell=0}^{\infty}(1-\alpha)^\ell M^\ell = \alpha(I-(1-\alpha)M)^{-1}.$$

.

The *PageRank vector* π is then the average of the personalized PageRank vectors:

$$\pi = \frac{1}{n}\sum_{u\in V}\pi_u = \frac{\mathbf{1}^T}{n}M_\alpha. \tag{5}$$

We immediately recover the well-known recurrence relation for π:

$$\pi = \frac{\alpha}{n}\mathbf{1}^T + (1-\alpha)\pi M. \tag{6}$$

Alternately, consider the column vector $\pi^{(\ell)}(\cdot,v)$, called the *ℓ-hop single-node PageRank vector* of v. It represents the probability of an ℓ-length α-random walk starting at each vertex ending at v. Note that $\pi^{(\ell)}(\cdot,v)$ is not a probability vector.

We can compute $\pi^{(\ell)}(\cdot,v) = \pi^{(\ell)}\mathbf{1}_v$. The *single-node PageRank vector* $\pi(\cdot,v)$ is defined by

$$\pi(\cdot,v) = \sum_{\ell=0}^{\infty}\pi^{(\ell)}(\cdot,v) = M_\alpha\mathbf{1}_v.$$

The *single-node PageRank value* $\pi(v)$ of v is then

$$\pi(v) = \frac{1}{n}\sum_{u\in V}\pi(u,v) = \frac{\mathbf{1}^T}{n}M_\alpha\mathbf{1}_v = \pi^T\cdot\mathbf{1}_v. \tag{7}$$

The last equality tells us that the single-node PageRank of v is actually simply v's entry in the PageRank vector - and hence the name.

Notice the following relation between the personalized and single-node PageRank vectors for undirected graphs: $\pi_u(v)d_u = \pi(v,u)d_v$, where d_u is the degree of u in G.

We also have the following useful reversibility result for undirected graphs:

$$\pi_u(v)d_u = \pi_v(u)d_v \tag{8}$$

2 Berkhin's Paint-Splitting Technique

There is an interesting *local* way to estimate PageRank vectors, which is invaluable on graphs where at-least-linear-time algorithms are infeasible. We shall call this the *paint-splitting* technique, after the paper of Berkhin [Ber05].

We shall need the following notation: define $\mathrm{pr}(\alpha, s) = s^T M_\alpha$. We say that p approximates $\mathrm{pr}(\alpha, s)$ to r if

$$p + \mathrm{pr}(\alpha, r) = \mathrm{pr}(\alpha, s). \tag{9}$$

The key point is the following: the *local* operation `push` preserves equation 9.

Algorithm 1 `push`(u)

$p(u) \leftarrow p(u) + \alpha r(u)$
for $v \in \mathrm{IN}(u)$ **do**
  $r(v) \leftarrow r(v) + \frac{(1-\alpha)}{d_u} r(u)$
end for
$r(u) \leftarrow 0$

Supposing p', r' are the vectors after `push`(u), we have

$$p'(v) + \mathrm{pr}(\alpha, r') = p(v) + \mathrm{pr}(\alpha, r) = \mathrm{pr}(\alpha, s). \tag{10}$$

One also sees that $\|r'\|_1 = \|r\|_1 - \alpha r(u) < \|r\|_1$ as long as $r(u) > 0$. Thus, starting with $p = 0$ and $r = s$, we can use the *local* `push` to iteratively approximate $\mathrm{pr}(\alpha, s)$ by p. Choosing vertices u cleverly could lead to fast approximation algorithms, as we see next.

3 Computing Personalized PageRank: ACL

Here, the authors note that the personalized PageRank vector π_u is simply $\mathrm{pr}(\alpha, \mathbb{1}_{[u]})$, and apply Berkhin repeatedly.

Fix $\epsilon > 0$. A vertex u is *active* if $r(u) \geq d_u \epsilon$. The ACL algorithm calls `push` on any active vertex u, till there are none left. At each stage, $\|r\|_1$ decreases by at least $\alpha \epsilon d_u$, and initially $\|r\|_1 = \left\|\mathbb{1}_{[u]}\right\|_1 = 1$, so that the total time $T = \sum_{\text{pushes vertices } u} d_u \leq \frac{1}{\alpha\epsilon}$.

The approximation computed leaves a residue vector r satisfying $r < \epsilon$ elementwise.

4 Computing Single-Node PageRank: Lofgren-Goel

Here we solve the "column" problem - computing $\pi(\cdot, v)$ for a fixed v - from which one can easily compute the single-node PageRank value $\pi(v)$.

The idea is similar to Berkhin's `push` algorithm, with the variables modified suitably for the column vector instead of the row vector. One nice contribution is the proof technique, where they use a binning-style idea to bound the running time. We shall not go into the details here.

The algorithm produces an approximation that is at most ϵ-far from the true vector $\pi(\cdot, v)$. The average-case running time is $\mathcal{O}(\frac{1}{\alpha\epsilon}\frac{m}{n})$.

5 Even faster: SetPush

There are two key problems with the algorithms till recently:

1. Algorithms that used random walks to estimate PageRank would need $\Omega(1/\pi(t))$ time to estimate $\pi(t)$, which could become $\Omega(n)$.

2. Other algorithms used `push`-style operations, which would also be $\Omega(n)$ in the case where the pushed vertex had degree $\Omega(n)$.

We solve the first problem by introducing *truncated PageRank vectors*, that only comprised $\mathcal{O}(\log n)$-length walks. The second is solved by interpolating between pushing mass to all neighbours and pushing mass to just a select set of neighbours, based on the amount of mass available and the degree of the vertex being pushed.

The algorithm produces a (c, ρ_f) approximation $\hat{\pi}(t)$ to the single-node PageRank $\pi(t)$: that is, an estimate $\hat{\pi}(t)$ satisfying $|\hat{\pi}(t) - \pi(t)| \leq c\pi(t)$ with probability at least $1 - \rho_f$. Note that c and ρ_f are constants here.

5.1 Truncated PageRank Vectors

Set $L = \log_{1-\alpha}\left(\frac{c\alpha}{2n}\right)$. We shall call L the *truncation length*. We define the *truncated PageRank vector* $\tilde{\pi}(t)$ to be the vector $\pi(t)$, but with all walks of length greater than L truncated. That is,

$$\tilde{\pi}(t) = \frac{1}{n} \sum_{s \in V} \sum_{\ell=0}^{L} \pi_s^{(\ell)}(t). \tag{11}$$

We could expect that long walks, thanks to the $(1-\alpha)^\ell$ factor, contribute very little to the PageRank vector. Indeed, this is the case, as the next lemma shows.

Lemma 2. *For any vertex v,*

$$|\pi(t) - \tilde{\pi}(t)| \leq \frac{c}{2}\pi(t).$$

It follows that $\hat{\pi}(t)$ is a (c, ρ_f) approximation to $\pi(t)$ if

$$|\hat{\pi}(t) - \tilde{\pi}(t)| \leq \frac{c}{2}\pi(t)$$

holds with probability at least $1 - \rho_f$.

Proof. We have

$$\begin{aligned}
|\pi(t) - \tilde{\pi}(t)| &= \frac{1}{n} \sum_{s \in V} \sum_{\ell=L+1}^{\infty} \pi_s^{(\ell)}(t) \\
&\leq \alpha \sum_{\ell=L+1}^{\infty} (1-\alpha)^\ell \left(\frac{1}{n} \sum_{s \in V} \mathbb{1}_{[s]}^T M^\ell \mathbb{1}_{[t]} \right) \\
&\leq \alpha \sum_{\ell=L+1}^{\infty} (1-\alpha)^\ell \\
&\leq (1-\alpha)^L = \frac{c\alpha}{2n} \\
&\leq \frac{c}{2}\pi(t).
\end{aligned}$$

□

5.2 The algorithm

Noting that

$$\tilde{\pi}(t) = \frac{1}{n} \sum_{s \in V} \sum_{\ell=0}^{L} \pi_s^{(\ell)}(t) = \frac{1}{n} \sum_{s \in V} \sum_{\ell=0}^{L} \frac{d_t}{d_s} \pi_t^{(\ell)}(s), \tag{12}$$

we compute approximations $\hat{\pi}_t^{(\ell)}(s)$ to each $\pi_t^{(\ell)}(s)$, and then use the above equation to compute $\hat{\pi}(t)$.

The algorithm is as follows:

Algorithm 2 The SetPush algorithm

Initialize two n-dimensional vectors $r_t^{(0)} \leftarrow \mathbb{1}_{[t]}$ and $\hat{\pi}_t^{(0)} \leftarrow \alpha \mathbb{1}_{[t]}$.
$L \leftarrow \log_{1-\alpha}\left(\frac{c\alpha}{2n}\right)$
for $\ell = 0$ to L **do** Initialize n-dimensional vector $r_t^{(\ell+1)} \leftarrow 0$.
  for each $u \in V$ with nonzero $r_t^{(\ell)}(u)$ **do**
    $p \leftarrow \frac{(1-\alpha) r_t^{(\ell)}(u)}{\theta d_u}$
    if $p \geq 1$ **then**
      for each $v \in N(u)$ **do**
        $r_t^{(\ell+1)}(v) \leftarrow r_t^{(\ell+1)}(v) + \theta p$
      end for
    else
      $idx \leftarrow 0$
      while true **do**
        $g \leftarrow \text{Geom}(p)$
        $idx \leftarrow idx + g$
        if $idx \geq d_u$ **then**
          break
        end if
        $r_t^{(\ell+1)}(N(u)[idx]) \leftarrow r_t^{(\ell+1)}(N(u)[idx]) + \theta$
      end while
    end if
  end for
  Initialize n-dimensional vector $\hat{\pi}_t^{(\ell+1)} \leftarrow \alpha r_t^{(\ell+1)}$
end for
$\hat{\pi}(t) \leftarrow \frac{1}{n} \sum_{s \in V} \sum_{\ell=0}^{L} \frac{d_t}{d_s} \hat{\pi}_t^{(\ell)}(s)$
return $\hat{\pi}(t)$

5.3 Analysis

Notice that the algorithm is randomized, and that the geometric random variable subroutine essentially simulates a Binomial(d_u, p) random variable with the advantage of taking $\mathcal{O}(p d_u)$ time instead of $\mathcal{O}(d_u)$ time, on average.

In the analysis, we first prove that the algorithm is correct in expectation, and then bound its variance. Finally, we compute a bound on the running time in terms of θ, and use Chebyshev's inequality to find a good value for θ.

Theorem 3. *The algorithm is correct in expectation:* $\mathbb{E}[\hat{\pi}(t)] = \pi(t)$.

Proof. We shall make use of the following random variables (notice that these are the only random variables in the algorithm): Let $X^{(\ell+1)}(u, v)$ denote the increment of $r_t^{(\ell+1)}(v)$ due to the update procedure at vertex u with nonzero $r_t^{(\ell)}(u)$.

The proof is actually quite simple: We show the following: for each $v \in V$ and $\ell \in \{0, 1, 2, \ldots, L\}$,

$$\mathbb{E}\left[r_t^{(\ell)}(v)\right] = \frac{1}{\alpha} \pi_t^{(\ell)}(v). \tag{13}$$

We induct on ℓ. The case $\ell = 0$ follows from the initialization of the algorithm. It is easy to see that the expectation of each $X^{(\ell+1)}(u, v)$ is given by

$$\mathbb{E}\left[X^{(\ell+1)}(u, v) \middle| r_t^{(\ell)}\right] = \theta p = \frac{(1-\alpha) r_t^{(\ell)}(u)}{d_u},$$

so that the expectation we need is

$$\mathbb{E}\left[\mathbb{E}\left[r_t^{(\ell+1)}(v) \middle| r_t^{(l)}\right]\right] = \mathbb{E}\left[\sum_{u \in V} \frac{(1-\alpha) r_t^{(\ell)}(u)}{d_u}\right] = \frac{1}{\alpha} \sum_{u \in V} \frac{(1-\alpha) \pi_t^{(\ell)}(u)}{d_u} = \frac{1}{\alpha} \pi_t^{(\ell+1)}(v),$$

where the last equality follows from Lemma 1. The theorem is then proved using linearity of expectation and Equation 12. □

We next bound the variance of the algorithm. We shall need a technical lemma that we shall not prove here, the proof is presented in the Appendix of [WW23].

Lemma 4. *For any vertex v and $\ell \in \{0, 1, 2, \ldots, L\}$,*

$$\mathrm{Var}\left[\hat{\pi}(t)\right] = \frac{\alpha^2}{n^2} \sum_{\ell=1}^{L-1} \mathbb{E}\left[\mathrm{Var}\left[\sum_{v \in V} r_t^{(\ell)}(v) \left(\sum_{s \in V} \frac{d_t}{d_s} \sum_{i=0}^{L-\ell} \frac{\pi_v^{(i)}(s)}{\alpha}\right) \middle| r_t^{(\ell-1)}\right]\right].$$

Using this, we now prove the key variance bound.

Theorem 5. *The algorithm has variance*

$$\mathrm{Var}\left[\hat{\pi}(t)\right] \leq \frac{L\theta d_t}{n} \cdot \pi(t).$$

Proof. We start from the expression in Lemma 4. Since the $X^{(\ell+1)}(u, v)$s are independent, we can move the variance operator into the summation over $v \in V$, and then use that $\left(\sum_{s \in V} \frac{d_t}{d_s} \sum_{i=0}^{L-\ell} \frac{\pi_v^{(i)}(s)}{\alpha}\right)$ is simply a constant now. Moving the expectation into the innermost summation, we now need to compute (the expectation of)

$$\mathrm{Var}\left[r_t^{(\ell)}(v) \middle| r_t^{(\ell-1)}\right] = \mathrm{Var}\left[\sum_{u \in N(v)} X^{(\ell)}(u, v) \middle| r_t^{(\ell-1)}\right] = \sum_{u \in N(v)} \mathrm{Var}\left[X^{(\ell)}(u, v) \middle| r_t^{(\ell-1)}\right].$$

The last term is 0 if p (defined in the algorithm) is at least 1, and is $\theta^2 p(1-p)$ otherwise. In any case, it is less than $\theta^2 p$ (perhaps there is some room for improving the upper bound in the case when p is large). Thus,

$$\mathbb{E}\left[\mathrm{Var}\left[r_t^{(\ell)}(v) \middle| r_t^{(\ell-1)}\right]\right] \leq \theta \sum_{u \in N(v)} \frac{(1-\alpha)\mathbb{E}\left[r_t^{(\ell-1)}(u)\right]}{d_u} = \frac{\theta}{\alpha} \pi_t^{(\ell)}(v).$$

We have so far:

$$\mathrm{Var}\left[\hat{\pi}(t)\right] \leq \frac{\alpha\theta}{n^2} \sum_{\ell=1}^{L-1} \sum_{v \in V} \left(\sum_{s \in V} \frac{d_t}{d_s} \sum_{i=0}^{L-\ell} \frac{\pi_v^{(i)}(s)}{\alpha}\right)^2 \pi_t^{(\ell)}(v).$$

We still have the term $\pi_v^{(i)}(s)$ term to deal with. However, Wang-Wei cleverly notice the following identity to get rid of them (which follows from the matrix forms of the expressions involved followed by an explansion of matrix multiplication):

$$\sum_{v \in V} \frac{1}{\alpha} \pi_v^{(i)}(s) \pi_t^{(\ell)}(v) = \frac{1}{\alpha} \pi_t^{(i+\ell)}(s). \tag{14}$$

This leaves us to remove the other term in the square. Here, Wang-Wei (if I might say, a little loosely) bound the term like so:

$$\sum_{s \in V} \frac{d_t}{d_s} \sum_{i=0}^{L-\ell} \frac{\pi_v^{(i)}(s)}{\alpha} \leq \frac{d_t}{\alpha} \sum_{s \in V} \sum_{i=0}^{L-\ell} \frac{\pi_v^{(i)}(s)}{1} \leq \frac{d_t}{\alpha} \sum_{s \in V} \pi_v(s) = \frac{d_t}{\alpha}.$$

Possible loose ends: substituting 1 for the average reciprocal degree could be improved, also, for ℓ close to L, the second inequality may not be quite tight. However, this might not matter at all, if there is no improvement obtained in the running time upon incorporating the optimizations.

We are ready to finish. We have

$$\begin{aligned}\operatorname{Var}[\hat{\pi}(t)] &\leq \frac{\theta d_t}{n^2} \sum_{\ell=1}^{L-1} \sum_{s \in V} \frac{d_t}{d_s} \sum_{i=0}^{L-\ell} \left(\sum_{v \in V} \frac{1}{\alpha} \pi_v^{(i)}(s) \pi_t^{(\ell)}(v) \right) \\ &\leq \frac{\theta d_t}{n^2} \sum_{\ell=1}^{L-1} \sum_{s \in V} \frac{d_t}{d_s} \pi_t(s) = \frac{\theta d_t}{n^2} \sum_{\ell=1}^{L-1} \sum_{s \in V} \pi_s(t) \\ &\leq \frac{\theta d_t}{n} \sum_{\ell=1}^{L-1} \pi(t) \leq \frac{L \theta d_t}{n} \pi(t),\end{aligned}$$

as needed. □

References

[ACL06] Reid Andersen, Fan Chung, and Kevin Lang. *Local Graph Partitioning using PageRank Vectors*. 2006. URL: https://ieeexplore.ieee.org/document/4031383.

[Ber05] Pavel Berkhin. *Bookmark-coloring algorithm for personalized pagerank computing*. 2005. URL: https://www.cs.cmu.edu/~15451-f20/Handouts/Berkhin-Painting.pdf.

[LG14] Peter Lofgren and Ashish Goel. *Personalized PageRank to a Target Node*. 2014. arXiv: 1304.4658 [cs.DS].

[WW23] Hanzhi Wang and Zhewei Wei. *Estimating Single-Node PageRank in* $\tilde{O}\left(\min\{d_t, \sqrt{m}\}\right)$ *Time*. 2023. arXiv: 2307.13162 [cs.DS].