Improved Communication Cost in Distributed PageRank Computation – A Theoretical Study

Siqiang Luo [1]

Abstract

PageRank is a widely used approach for measuring the importance of a node in a graph. Due to the rapid growth of the graph size in the real world, the importance of computing PageRanks in a distributed environment has been increasingly recognized. However, only a few previous works can provide a provable complexity and accuracy for distributed PageRank computation. Given a constant $d \geq 1$ and a graph of n nodes, the state-of-the-art approach, Radar-Push, uses $O(\log \log n + \log d)$ communication rounds to approximate the PageRanks within a relative error $\Theta(\frac{1}{\log^d n})$ under a generalized congested clique distributed computation model. However, Radar-Push entails as large as $O(\log^{2d+3} n)$ bits of bandwidth (e.g., the communication cost between a pair of nodes per round). In this paper, we provide a new algorithm that uses asymptotically the same communication round complexity while using only $O(d \log^3 n)$ bits of bandwidth.

1. Introduction

A graph is used to model a set of instances (represented as "nodes") with the mutual relationship (represented as "edges") between some pairs of instances. For example, a social network can be modeled as a graph where a node is a user, and an edge indicates friendship (Fang et al., 2016); a road network is also a graph with a node as a road junction and an edge as a road segment (Luo et al., 2018; 2019a). In these graphs, PageRank (Page et al., 1999) is an effective approach in estimating the importance of the nodes, and has been applied in numerous applications including data mining, machine learning, distributed networks as well as Web algorithms (Luo et al., 2019b;c; Das Sarma et al., 2015; Florescu & Caragea, 2017; Fujiwara et al., 2013; Ahmadi et al., 2011; Neumann et al., 2011; Ponzetto & Strube, 2007). Given a graph G of n nodes and a probability $\alpha \in (0, 1)$, the PageRank vector of G is the stationary distribution π of the following specific random walk with restart: a walker initially starts at a node chosen uniformly at random; at each step, with probability $1 - \alpha$ the walker jumps to a neighboring node chosen randomly, and with the remaining α probability it restarts at a random node.

Computing PageRanks in a distributed environment has been extensively studied (Luo, 2019; Guo et al., 2017; Das Sarma et al., 2015; Zhu et al., 2005). Among them, IPRA [1] (Das Sarma et al., 2015) and Radar-Push (Luo, 2019) provide provable complexity and accuracy in a vertex-centric distributed computation model. Particularly, IPRA is discussed under a communication-restricted variant congested clique that initially each node only knows its neighbors, whereas Radar-Push employs a bandwidth-generalized congested clique model, which considers each graph node as individuals and that the nodes communicate with each other over a complete network (i.e., the clique on the n nodes of the graph) in order to compute some function of their inputs. The computation is performed in synchronous communication rounds and each pair of nodes can communicate at most b bits per round, where b is known as the bandwidth of the model. Radar-Push assumes that b can be $\Omega(\log n)$, where n is the number of nodes in the graph. Under the model, Radar-Push is the state-of-the-art algorithm that gives the best round complexity. For an undirected and connected graph G of n nodes, Radar-Push approximates PageRank vector with a relative error of $\Theta(\frac{1}{\log^d n})$ in $O(d \log \log n)$ communication rounds with a bandwidth of $O(\log^{2d+3} n)$ bits.

The congested clique model is a fundamental distributed computation model, and it has received tremendous interest in recent years. Under this model many important problems have been studied. Examples include computing graph connectivity (Hegeman et al., 2015), computing Minimum Spanning Tree (Ghaffari & Parter, 2016; Jurdziński & Nowicki, 2018), as well as addressing the routing and sorting problem (Lenzen, 2013). The congested clique model is

[1] Harvard University. Correspondence to: Siqiang Luo <siqiangluo@seas.harvard.edu>.

Proceedings of the 37^{th} International Conference on Machine Learning, Online, PMLR 119, 2020. Copyright 2020 by the author(s).

[1] Named "Improved-PageRank-Algorithm" in the original paper.

fundamental to other important distributed models such as the k-machine model (Klauck et al., 2014) and MapReduce model (Hegeman & Pemmaraju, 2015). As shown in (Klauck et al., 2014; Hegeman & Pemmaraju, 2015), an efficient algorithm designed for the congested clique model can typically inspire an efficient algorithm on the k-machine model and MapReduce model. In this paper, we generalize the standard congested clique model in that we allow the bandwidth to go beyond $O(\log n)$, following the trend of discussing the tradeoff between round complexity and bandwidth-per-round (Beame et al., 2017). Also, discussing a more flexible bandwidth can be interesting for modern vertex-centric computation systems (Malewicz et al., 2010) that employ node-to-node communication mechanism similar to the congested clique model, because the bandwidth between node pairs can be translated to the maximum allowed communication between machines.

Our Contributions. We present an improved distributed PageRank algorithm based on the bandwidth-generalized congested clique model. In particular, given a constant $d \geq 1$, and a graph G (connected and undirected) of n nodes, our algorithm approximates the PageRank values of nodes in G within a relative error of $\Theta(\frac{1}{\log^d n})$ for any node with a probability at least $1 - \frac{1}{n}$. Furthermore, our algorithm finishes the computation in $O(d \log \log n)$ communication rounds with a bandwidth of $O(d \log^3 n)$ bits in the worst case. Our algorithm significantly improves the state of the art, Radar-Push, which incurs a bandwidth of $O(\log^{2d+3} n)$ bits.

2. Preliminaries

In this section we present the concepts and definitions that are required to discuss the algorithm.

Bandwidth-Generalized Congested Clique Model. The congested clique model is widely used in analyzing the efficiency of distributed algorithms. Given an undirected and connected simple graph $G(V, E)$, where V is the node set and E is the edge set, we consider the clique $G'(V, E')$, which is a complete overlay network based on the node set V. The overlay network is regarded to be the communication network that any pair of nodes can communicate with each other. The nodes have unique identifiers 1 to n that are known to all other nodes. The communication is conducted in synchronized rounds. At each round, each node receives messages from other nodes, performs arbitrary computations, and sends a message to other nodes. In particular, if a message has been sent in round i from a node u to v, then node v would receive the message when round $i + 1$ starts. Each node can perform unlimited computation. The model also contains a bandwidth parameter b such that each pair of nodes (u, v) can exchange at most messages of b bits in each round. The standard congested clique model

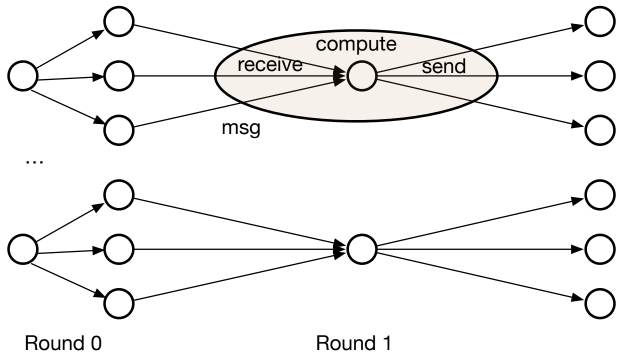

Figure 1. Illustrating the communication rounds in the model.

often assumes $b = O(\log n)$. In contrast, in this paper, we allow $b = \Omega(\log n)$. At each round, each node of graph G conducts local computation with the received messages, and the task is to solve a graph problem (e.g., PageRank computation) related to G. An example computation in the model is shown in Figure 1. Initially (in Round 0), each node can communicate with some other nodes by sending them messages. At the beginning of Round 1, the messages that were sent by other nodes in Round 0 reach their destinations, which are then processed locally at the destination nodes. This process repeats similarly in the following rounds. In this model, we focus on two important measures, which are the round complexity (i.e., the number of rounds with regard to n) and the bandwidth b.

α-Decay Random Walk. Given a graph $G(V, E)$ and a constant $\alpha \in (0, 1)$, an α-decay random walk (typically starting from a certain node s chosen uniformly at random) is a traversal on G such that at each step, it stops at the current node with probability α; with the remaining $1 - \alpha$ probability, it moves to a neighboring node chosen uniformly at random. Note that the complexities we give later may hide an α related factor as we treat α as a constant.

Length-L Random Walk. A length-L random walk is a 0-decay random walk but stops immediately after moving L steps. By definition, generating α-decay random walks is equal to generating length-L random walks where L is drawn from a geometry distribution Geom(α). This transforms generating α-decay random walks to fixed-length random walks.

The Problem of PageRank Approximation. Given an error bound ϵ, and a failure probability p_f, an approximate PageRank query returns an estimated PageRank $\hat{\pi}(v)$ for every node v, such that

$$|\pi(v) - \hat{\pi}(v)| \leq \epsilon \cdot \pi(v) \quad (1)$$

holds with a probability at least $1 - p_f$. Here, $\pi(v)$ denotes the actual PageRank value of node v. Following the typical definition of "high probability", we set $p_f = \frac{1}{n}$. In this paper, we are interested in the cases where $\epsilon = \Theta(\frac{1}{\log^d n})$ for a constant $d \geq 1$.

Our Goal: Distributed PageRank Approximation. We aim to design distributed algorithms based on the bandwidth-generalized congested clique model to approximate the PageRanks of nodes in G, with the approximation guarantee shown in Equation 1.

3. Main Competitors

IPRA (Das Sarma et al., 2015) and Radar-Push (Luo, 2019) are the closest works that also study the distributed computation of PageRank in a vertex-centric model, and these two algorithms can be applied in the model in this paper.

IPRA. Let $\pi(u)$ be the PageRank value of node u and n be the number of nodes in the graph. IPRA approximates the PageRank values by simulating a number of α-decay random walks. Each α-decay random walk starts from a node uniformly chosen at random, and with a probability $\alpha \in (0, 1)$ it stops at the current node, while with a probability $1 - \alpha$ it jumps to a random neighbor. It can be shown [2] that if we simulate $K = O(n \log^{2d+1} n)$ random walks and K_u of them stop at node u, then with a probability at least $1 - \frac{1}{n}$, $\hat{\pi}(u) = \frac{K_u}{K}$ is an unbiased estimate of $\pi(u)$ within a $\Theta(\frac{1}{\log^d n})$ relative error. Directly using such random walk simulation method will achieve, with a probability at least $1 - \frac{1}{n}$, a round complexity of $O(\frac{\log n}{\alpha}) = O(\log n)$ since the longest random walk is of length $O(\frac{\log n}{\alpha})$ with high probability. To improve the round complexity to $O(\sqrt{\log n})$, the IPRA algorithm employs a two-phase approach. The first phase spends $O(\sqrt{\log n})$ rounds to compute some random walks of length $O(\sqrt{\log n})$ from each node. The second phase, using $O(\sqrt{\log n})$ rounds, computes every random walk of length $O(\sqrt{\log n})$ by stitching $O(\sqrt{\log n})$ precomputed random walks in the first phase. The bandwidth of the algorithm can be shown to be $O(\log^{2d+3} n)$ bits. We note that IPRA's model imposes a constraint that initially each node only communicates with its neighboring nodes, whereas in this paper we assume each node knows all the node IDs initially.

Radar-Push. The Radar-Push algorithm presented in (Luo, 2019) significantly improves the round complexity, giving a round complexity of $O(\log \log n)$ rounds with a bandwidth of $O(\log^{2d+3} n)$ bits. Radar-Push achieves better round complexity by recursively stitching the shorter walks to generate a longer random walk with a doubled length.

[2]By the Chernoff bound, it can be shown that using $\frac{6 \log(2n)}{\epsilon^2 \alpha} = O(\frac{n \log n}{\epsilon^2})$ random walks gives us a relative estimate error ϵ (see (Luo, 2019) for more details). We then can easily extend the analysis by setting $\epsilon = \Theta(\frac{1}{\log^d n})$ and obtain the number of walks as $K = O(n \log^{2d+1} n)$.

Algorithm 1 Estimating PageRanks based on Unit Task

```
$\epsilon^* \leftarrow \frac{\epsilon}{2}$, $k \leftarrow \frac{6\log(2n)}{\epsilon^{*2}\alpha}$
for $u \in V$ do
    $\hat{\pi}(u) \leftarrow 0$
end
Parallel for $L = 0$ to $\log_{\frac{1}{1-\alpha}}(\frac{1}{\epsilon^*} \cdot \frac{n}{\alpha})$ do
    invoke $\lceil k \cdot (1-\alpha)^L \cdot \alpha \rceil$ unit tasks for length $L$ simultaneously
    for each walk from node $v$ to node $u$ do
        $\hat{\pi}(u) \leftarrow \hat{\pi}(u) + \frac{k \cdot (1-\alpha)^L \cdot \alpha}{n \cdot k \cdot d(v) \cdot \lceil k \cdot (1-\alpha)^L \cdot \alpha \rceil}$
    end
end
```

For example, a length-16 random walk can be computed by connecting two length-8 random walks, each can then be connected by two length-4 random walks, and so on. Based on this process, generating a length-L walk requires $O(\log L)$ rounds. Radar-Push gives a new PageRank estimator that is based on the unit task, which is defined as follows.

Definition 1. *A unit task for length L is about generating $d(u)$ (the degree of node u) length-L random walks from each node u.*

The algorithm that estimates PageRanks based on the unit task is described in Algorithm 1, which has been shown to give a desirable relative error guarantee in terms of PageRank estimation. To generate k random walks, the idea of Algorithm 1 is to accomplish $k \cdot (1-\alpha)^L \cdot \alpha$ length-L unit tasks for L, as a length-L α-decay random walk happens with probability $(1-\alpha)^L \cdot \alpha$. Note that it is sufficient to only consider $L \in [0, \log_{\frac{1}{1-\alpha}}(\frac{1}{\epsilon^*} \cdot \frac{n}{\alpha})]$ for a desirable accuracy, since $\sum_{L \in [0, \log_{\frac{1}{1-\alpha}}(\frac{1}{\epsilon^*} \cdot \frac{n}{\alpha})]} (1-\alpha)^L \alpha = 1 - \frac{\alpha \cdot (1-\alpha)}{n} \cdot \epsilon^*$. This indicates that a significant $1 - \frac{\alpha \cdot (1-\alpha)}{n} \cdot \epsilon^*$ portion of unit tasks have been conducted if we discard the range $L \in [\log_{\frac{1}{1-\alpha}}(\frac{1}{\epsilon^*} \cdot \frac{n}{\alpha}), \infty]$. We refer interesting readers to the appendix for more details on why this cut-off is acceptable.

The Reason of Using Unit Tasks. One major challenge addressed by Radar-Push is evaluating the number of shorter random walks that are required to be stitched for the longer ones. In particular, Radar-Push employs a shuffle-and-send steps for simultaneously generating the random walks from each node, and it has been shown that the random walks such generated will not worsen the PageRank estimation. When multiple random walks reach a node u during the random walk process, the shuffle-and-send mechanism requires that the walks are extended to u's neighboring nodes as evenly as possible, in contrast to independently choosing random neighbors for each walks. As such, in a unit task, the random walks received and sent by node u at each round

Algorithm 2 Estimating PageRanks based on Simple Unit Task

```
$\epsilon^* \leftarrow \frac{\epsilon}{2}, k \leftarrow \frac{6\log(2n)}{\epsilon^{*2}\alpha}$
for $u \in V$ do
    $\hat{\pi}(u) \leftarrow 0$
end
Parallel for $L = 0$ to $\log_{\frac{1}{1-\alpha}}(\frac{1}{\epsilon^*} \cdot \frac{n}{\alpha})$ do
    invoke $\lceil k \cdot (1-\alpha)^L \cdot \alpha \rceil$ simple unit tasks for length $L$ simultaneously
    for each walk from node $v$ to node $u$ do
        $\hat{\pi}(u) \leftarrow \hat{\pi}(u) + \frac{k\cdot(1-\alpha)^L\cdot\alpha}{n\cdot k\cdot d(v)\cdot\lceil k\cdot(1-\alpha)^L\cdot\alpha\rceil}$
    end
end
```

is exactly $d(u)$. This property allows a precise prediction of how many shorter random walks need to be generated beforehand. Particularly, if we want to perform a unit task for length $2L$, then we can perform two unit tasks of length L and concatenate them together as there are exactly $d(u)$ length-L walks starting and ending at u.

The main result of Radar-Push is given as follows.

Lemma 1. *Radar-Push requires $O(\log L)$ rounds and a bandwidth of $O(\log^2 n)$ bits to compute a unit task for length-L walks.*

We use Lemma 1 to analyze the bandwidth of Algorithm 1. Assuming α is a constant and setting $\epsilon = \Theta(\frac{1}{\log^d n})$ (and therefore $\epsilon^* = \frac{\epsilon}{2} = \Theta(\frac{1}{\log^d n})$) bounds the number of unit tasks as follows.

$$\begin{aligned}
&\sum_{L\in[0,\log_{\frac{1}{1-\alpha}}(\frac{1}{\epsilon^*}\cdot\frac{n}{\alpha})]} \lceil k\cdot(1-\alpha)^L\cdot\alpha\rceil \\
\leq &\sum_{L\in[0,\log_{\frac{1}{1-\alpha}}(\frac{1}{\epsilon^*}\cdot\frac{n}{\alpha})]} (k\cdot(1-\alpha)^L\cdot\alpha+1) \\
\leq &k+1+\log_{\frac{1}{1-\alpha}}(\frac{1}{\epsilon^*}\cdot\frac{n}{\alpha}) \\
= &O(\log^{2d+1} n) \qquad (2)
\end{aligned}$$

Then, by Lemma 1 the bandwidth is $O(\log^{2d+1} n \cdot \log^2 n) = O(\log^{2d+3} n)$ bits. We can further show that in this case the round complexity is $O(\log\log n + \log d)$, and we discuss in detail in Appendix.

4. Our Idea: Fractional-Push + Radar-Push

We present a new algorithm that significantly improves the bandwidth from $O(\log^{2d+3} n)$ bits to $O(d\log^3 n)$ bits.

4.1. Main Algorithm and Correctness

We introduce two types of push steps for computing PageRank. One, as described in Section 3, is the shuffle-and-send operation in Radar-Push (in what follows, we will also refer to this operation as Radar-Push when the context is clear). Suppose there are k random walks currently reach at a node u, then Radar-Push will distribute the random walks to the neighbors of u as evenly as possible. For example, if $k = 10$ and the node u has a degree $d(u) = 4$, then there are 2 neighbors that will receive 2 walks from u and the other 2 neighbors will receive 3 walks. The other type of push is called Fractional-Push (also known as the forward push (Berkhin, 2006; Andersen et al., 2006)). The main idea of the Fractional-Push is to conceptually fractionalize the number of walks to be extended to each neighbor. In particular, it retains α portion of random walks at the current node, and for the remaining $1-\alpha$ portion it distributes to each of its neighbors the same number of walks. Using the aforementioned example and suppose $\alpha = 0.1$, it retains $10 \cdot 0.1 = 1$ random walks at the current node u and distributes each neighbors of u a number $\frac{9}{4}(= 2.25)$ of random walks. Although it is practically impossible to conduct 0.25 random walks, we can regard 0.25 as a conceptual number without really doing the random walk simulations (namely, recording the number of walks staying/passing through at each node).

Our main idea is to combine these two types of pushes for PageRank computation, with the purpose of reducing the bandwidth while still giving the same theoretical guarantee in PageRank approximation. As shown in Figure 2, when doing multiple unit tasks in parallel, the bottleneck of bandwidth exists in the initial communication rounds. To explain, in these rounds, a large number of short random walks are created for later use and thus piling up the messages transmitted between nodes. This issue can be addressed by using Fractional-Push in the initial rounds. When doing t unit tasks simultaneously, for example, simply conducting Radar-Push costs $O(t\log^2 n)$ bits of bandwidth because each unit task requires $O(\log^2 n)$ bits of bandwidth. However, if in the first few rounds we use Fractional-Push, then at every round, it is sufficient for each pair of nodes only to communicate a float number that is the number of random walks distributed from one node to the other. Transmitting a float number can be relatively small compared with $O(t\log^2 n)$ bits, and we will give a more detailed analysis shortly. The main idea is illustrated in Figure 2. A thicker link between nodes refers to a higher communication cost. On the left of the figure, we illustrate parallel processing two unit tasks. In the first two rounds, the communication cost is the bottleneck. This communication cost is significantly reduced if Fraction-Push is applied.

Following this line of thought, there are several challenges.

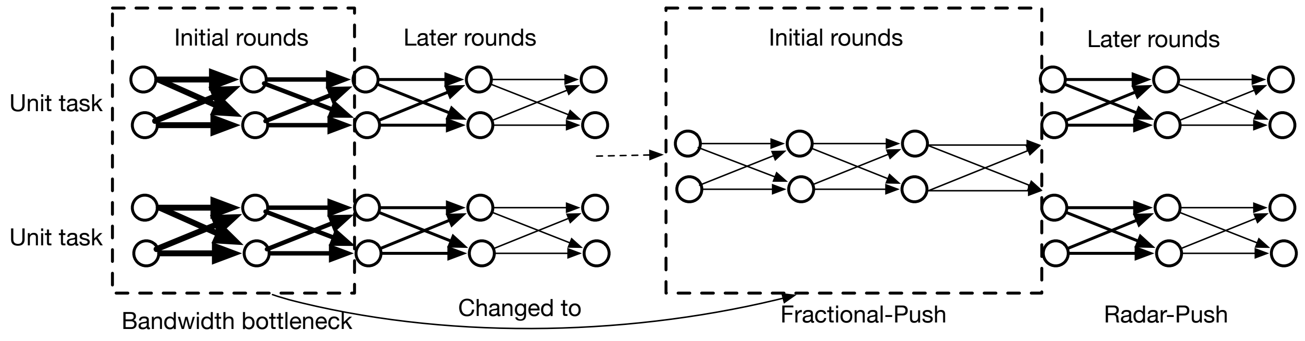

Figure 2. Illustrating the main idea of our algorithm.

First, based on Fractional-Push it is difficult to trace the source node of each walk which is required if the estimation is based on the unit task; second, it is not clear how many rounds we should use for Fractional-Push and how many for Radar-Push, as well as how to combine them; third, encoding a float number always introduces rounding error and thus we need new analysis for estimating the number of bits required to encode a float number for an acceptable rounding error. Below we discuss in detail how we address the challenges.

Tracing Source Node is Not Necessary. We address the source-node trace problem by using the standard Monte-Carlo method for the Fractional-Push phase. This standard method does not require to trace the source nodes of the walks. Instead, it only simulates one walk from each node, and it has been shown to be sufficient to achieve the desirable PageRank estimation guarantee. In particular, if we define the following simple unit task (Definition 2) and replace it with the unit task in Algorithm 1, we still get the desirable guarantee as in Equation 1 as long as we change the estimation accordingly.

Definition 2. *(Simple Unit Task) A simple unit task for length-L walks simulates from each node a random walk of length L.*

The revised algorithm is given in Algorithm 2, with two modifications compared with Algorithm 1: 1) we use the simple unit task instead of the unit task; 2) we modify the estimation of the PageRank values. The estimation follows the classic Monte Carlo approach that approximates the PageRank of a node by the proportion of the walks that end at that node.

Combining Fractional-Push with Radar-Push. We modify Algorithm 2 as follows to combine the two types of pushes. In the first $\log k (= O(\log(\log^{2d+1} n)) = O(d \log\log n))$ rounds we use Fractional-Push that conducts the simple unit tasks. Since initially the total number of walks is $k \cdot n$ and Fractional-Push is α-decay, after $O(\log k)$ rounds the number of the active random walks (i.e., those walks have not stopped) remains to be only $O(n)$, and these walks will be conducted with Radar-Push. Our algorithm is described in Algorithm 3, which is divided into three phases: 1) Initialization Phase; 2) Fractional-Push Phase; and 3) Radar-Push Phase. In Initialization Phase, two types of variables, $\hat{\pi}$ and τ are initialized. $\hat{\pi}(u)$ records the portion of walks that have been terminated at node u. $\hat{\pi}(u)$ is progressively updated during the Fractional-Push phase and Radar-Push phase, and it is the final estimation of PageRank value of u when the algorithm terminates. $\tau(u)$ records the portion of walks that currently reside at node u but have not been terminated. We note that the idea of using these two types of variables has also been employed in (Andersen et al., 2006; Wang et al., 2017). Fractional-Push operations are described in Lines 5-14. Radar-Push phase (Lines 15-20) conducts multiple unit tasks and updates $\hat{\pi}$ values.

Correctness. We now prove that the estimation accumulated by Line 18 of Algorithm 3, in expectation, gives a desirable estimation of $\pi(u)$. We first give the following lemma (rephrased based on the results in (Andersen et al., 2006)).

Lemma 2. *At any state of the Fractional-Push phase of Algorithm 3, the PageRank of u can be computed by $\pi(u) = \hat{\pi}(u) + \sum_{v \in V} \tau(v) \cdot \pi(v,u)$, where $\pi(v,u)$ denotes the probability that an α-decay random walk starting from v ends at u.*

We apply Lemma 2 by letting $\hat{\pi}(u)$ be the values after the termination of Fractional-Push phase. The computation of $\pi(u)$ in Algorithm 3 is divided into two parts. The first part, $\hat{\pi}(u)$, is computed by the Fractional-Push phase of Algorithm 3 (Lines 5-14), and the second part is estimated by the Radar-Push phase in Algorithm 3 (Lines 15-20). Hence, it is sufficient to show that the increment of $\hat{\pi}(u)$ by Line 18 sums up to $\sum_{v \in V} \tau(v) \cdot \pi(v,u)$. Let us fix two nodes v and u, and let $I_\alpha(v,u)$ be the indicator of the event that an α-decay random walk that starts from v ends at u. Further, let $I'_L(v,u)$ be the indicator of the event that

Algorithm 3 Estimating PageRanks with Improved Bandwidth

```
/* Initialization Phase */
1  $\epsilon \leftarrow \frac{1}{\log^d n}$    $k \leftarrow \frac{(2\epsilon/3+2)\log(2n)}{\epsilon^2\alpha}$
2  for $u \in V$ do
3      $\tau(u) \leftarrow \frac{1}{n}$   $\hat{\pi}(u) \leftarrow 0$
4  end
/* Fractional-Push Phase */
5  for $L = 1$ to $\log_{\frac{1}{1-\alpha}} k$ /* One round per loop */
6  do
7      for $u \in V$ do
8          $\hat{\pi}(u) \leftarrow \hat{\pi}(u) + \tau(u)\cdot\alpha$
9          for $v$ that is a neighbor of $u$ do
10             $\tau(v) \leftarrow \tau(v) + (1-\alpha)\cdot\frac{\tau(u)}{d(u)}$
11         end
12         $\tau(u) \leftarrow 0$
13     end
14 end
/* Radar-Push Phase */
15 Parallel for $L = 0$ to $2d\log_{\frac{1}{1-\alpha}} n$ do
16     invoke $\lceil(1-\alpha)^L\cdot\alpha\rceil$ unit tasks for length-$L$ simultaneously
17     for each walk from node $v$ to node $u$ do
18         $\hat{\pi}(u) \leftarrow \hat{\pi}(u) + \tau(v)\cdot\frac{(1-\alpha)^L\cdot\alpha}{d(v)\cdot\lceil(1-\alpha)^L\cdot\alpha\rceil}$
19     end
20 end
```

a length-L walk that starts from v terminates at u. Also, we let $\Pr[L, v, u]$ be the probability that a length-L random walk that starts from node v ends at u. We have

$$\begin{aligned}
&\sum_{v\in V}\tau(v)\cdot\pi(v,u) = \sum_{v\in V}\tau(v)\mathbb{E}[I_\alpha(v,u)] \\
&= \sum_{v\in V}\tau(v)\mathbb{E}_{L\sim Geom(\alpha)}[I'_L(v,u)] \\
&= \sum_{L\sim Geom(\alpha)}\sum_{v\in V}\tau(v)\Pr[L,v,u] \\
&= \sum_{L\geq 0}\sum_{v\in V}\tau(v)(1-\alpha)^L\alpha\Pr[L,v,u] \\
&= \underbrace{\sum_{L\in[0,2d\log_{\frac{1}{1-\alpha}} n]}\sum_{v\in V}\tau(v)(1-\alpha)^L\alpha\Pr[L,v,u]}_{\text{denoted as A}} + \\
&\underbrace{\sum_{L>2d\log_{\frac{1}{1-\alpha}} n}\sum_{v\in V}\tau(v)(1-\alpha)^L\alpha\Pr[L,v,u]}_{\text{denoted as B}} \qquad (3)
\end{aligned}$$

We show that part A in Equation 3 is unbiasedly estimated by the accumulated increment in Line 18 of Algorithm 3, and part B is an acceptable rounding error. Let us first focus on part A and see how $\Pr[L, v, u]$ can be estimated with the unit task for length L. Note that a unit task simulates $d(u)$ Length-L random walks from every node $u \in V$. If we let a length-L walk from v to u contribute $\frac{1}{d(v)}$ to estimating $\Pr[L, v, u]$, then $\Pr[L, v, u]$ can be unbiasedly estimated by incrementing $\frac{1}{d(v)}$ once a length-L walk from v to u is observed. One issue is that for each length-L unit tasks we round the number $(1-\alpha)^L\alpha$ to $\lceil(1-\alpha)^L\alpha\rceil$ because the number of unit tasks must be an integer. To achieve an unbiased estimation we need a rescale factor $\frac{(1-\alpha)^L\alpha}{\lceil(1-\alpha)^L\alpha\rceil}$ for the contribution from all the length-L walks. Since the probability of a length-L walk happens with probability $(1-\alpha)^L\alpha$, summing up $\tau(v)\cdot\frac{(1-\alpha)^L\alpha}{d(v)\cdot\lceil(1-\alpha)^L\alpha\rceil}$ for each observation of a length-L walk from v to u gives an unbiased estimation of $\sum_{v\in V}\tau(v)(1-\alpha)^L\alpha\Pr[L, v, u]$, and further summing over L forms part A.

As for part B, we have

$$\begin{aligned}
&\sum_{L>2d\log_{\frac{1}{1-\alpha}} n}\sum_{v\in V}\tau(v)(1-\alpha)^L\alpha\Pr[L,v,u] \\
&= \sum_{L>2d\log_{\frac{1}{1-\alpha}} n}(1-\alpha)^L\alpha\cdot\sum_{v\in V}\tau(v)\Pr[L,v,u] \\
&\leq \sum_{L>2d\log_{\frac{1}{1-\alpha}} n}(1-\alpha)^L\alpha\cdot\sum_{v\in V}\tau(v) \\
&\leq \sum_{L>2d\log_{\frac{1}{1-\alpha}} n}(1-\alpha)^L\alpha\cdot 1 \\
&\leq \frac{1}{n^{2d}} \qquad (4)
\end{aligned}$$

Since G is undirected and connected, $\pi(u)$ is at least $\frac{\alpha}{n}$ (consider that there is $\frac{1}{n}$ portion of α-decay walks starting at u and at least α portion of them stop at u), neglecting $O(\frac{1}{n^{2d}})$ does not hurt the $\Theta(\frac{1}{\log^d n})$ relative error guarantee for $d \geq 1$.

Accuracy. Further, we show that the estimation by Algorithm 3 gives the desirable guarantee shown in Equation 1. For this purpose, we first construct a relevant algorithm $\mathcal{A}$ which gives a desirable estimation of the PageRanks (i.e., satisfying Equation 1), and then show that Algorithm 3 gives at least the same accuracy guarantee as $\mathcal{A}$. $\mathcal{A}$ is constructed by replacing the Radar-Push phase in Algorithm 3 with pure α-decay random walks. As shown in (Luo, 2019), Radar-Push gives a better concentration than pure α-decay random walks in estimating PageRanks. Hence, it is sufficient to show that $\mathcal{A}$ gives the desirable accuracy. Our analysis is based on the following lemma.

Lemma 3. *In Algorithm 3, after the Fractional-Push phase,*

$\frac{(2\epsilon/3+2)n\log(2n)}{\epsilon^2\alpha}\cdot\tau(u)\le d(u)$ *for every node* $u\in V$.

Proof. Suppose initially we increase the initial value of $\tau(u)$ to be $\frac{d(u)}{n}$ (instead of $\frac{1}{n}$). Then, after the Fractional-Push phase, the τ values computed by the original algorithm should be at most those corresponding τ values computed by the modified algorithm (hereinafter we refer to as τ' values). Thus it is sufficient to show that after the Fractional-Push phase, $\frac{(2\epsilon/3+2)n\log(2n)}{\epsilon^2\alpha}\cdot\tau'(u)=d(u)$ holds for every node $u\in V$. To compute the τ' values after the Fractional-Push phase, we note that 1) initially $\sum_{v\in V}\tau'(v)=\frac{\sum_{v\in V}d(v)}{n}$; 2) each loop (Line 5) reduces $\sum_{v\in V}\tau'(v)$ by α portion. Let $k=\frac{(2\epsilon/3+2)\log(2n)}{\epsilon^2\alpha}$ and consider the case after the Fractional-Push phase, we have

$$\begin{aligned}&\sum_{v\in V}\frac{(2\epsilon/3+2)n\log(2n)}{\epsilon^2\alpha}\cdot\tau'(v)\\=&\sum_{v\in V}\frac{(2\epsilon/3+2)d(v)\log(2n)}{\epsilon^2\alpha}\cdot(1-\alpha)^{\log_{\frac{1}{1-\alpha}}k}\\=&\sum_{v\in V}d(v)\end{aligned}\tag{5}$$

As the initial values of τ' are degree proportional, $\frac{\tau'(v)}{d(v)}=\frac{\tau'(u)}{d(u)}$ holds for any two nodes v and u after each loop in Line 5. Hence, we have $\frac{(2\epsilon/3+2)n\log(2n)}{\epsilon^2\alpha}\cdot\tau'(v)=d(v)$ for every node $v\in V$ after the Fractional-Phase phase. This implies Lemma 3 holds. □

In the Radar-Push phase of Algorithm $\mathcal{A}$, a combination of performing $\lceil(1-\alpha)^L\cdot\alpha\rceil$ unit tasks for length L (for all L) gives a better PageRank concentration than directly doing 1 unit task, i.e., performing $d(u)$ α-decay random walks from each node u. Lemma 3 immediately implies that the Radar-Push phase would also give a better PageRank concentration than conducting $\frac{(2\epsilon/3+2)n\log(2n)}{\epsilon^2\alpha}\cdot\tau(u)$ α-decay random walks from each node u. To analyze the estimation accuracy of Algorithm $\mathcal{A}$, a natural question arises: *if we replace the Radar-Push phase with performing* $\frac{(2\epsilon/3+2)n\log(2n)}{\epsilon^2\alpha}\cdot\tau(u)$ *α-decay random walks from each node u (which only gives a weaker concentration than the original algorithm), can we achieve the desirable accuracy guarantee?* Let us denote the modified algorithm as $\mathcal{A}'$. Note that $\mathcal{A}'$ also needs to modify its PageRank estimation method for an unbiased estimate. That is, after the Fractional-Push phase, each walk that ends at node u contributes u' PageRank estimation, $\hat{\pi}(u)$, by $\frac{1}{kn}(=\frac{\epsilon^2\alpha}{(2\epsilon/3+2)n\log(2n)})$. Next, we show that Algorithm $\mathcal{A}'$ gives a desirable estimation.

Lemma 4. *With Algorithm $\mathcal{A}'$, $|\pi(v)-\hat{\pi}(v)|\le\epsilon\cdot\pi(v)$ holds with probability at least $1-\frac{1}{n}$.*

Proof. We first reduce the problem of computing PageRank in the original graph G to computing Personalized PageRank in a relevant graph G'. G' is constructed from G based on the following operations: 1) G' has an additional node s^* compared with G; 2) each undirected edge (u,v) in G makes two directed edges (u,v) and (v,u) in G'; 3) s^* links to every other node in G', i.e., there are directed edges (s^*,u) in G' for every other node u. We claim that the personalized PageRank from s^* to v in G' is the same as the PageRank of v in G. To see this, the personalized PageRank from source node s^* is defined as the probability of an α-decay random walk that starts from s^* terminates at v. Since s^* links to every other node (but is not linked reversely), the probability is equal to an α-decay random walk in G that starts from a node uniformly chosen at random, and the walk terminates at node v.

Thus, we can apply $\mathcal{A}'$ in G' to compute the personalized PageRank from s^* to other node v in G. Let us denote the personalized PageRank value as $\pi(s^*,v)$ and the approximate value by Algorithm $\mathcal{A}'$ is $\hat{\pi}(s^*,v)$. We note that Algorithm $\mathcal{A}'$ is a BATON instance (Luo et al., 2019c) and as shown in (Wang et al., 2017), the algorithm gives the following guarantee (lemma rephrased).

Lemma 5. *Given a value threshold $\delta\in(0,1)$, when setting $k=\frac{(2\epsilon/3+2)\log(2n)}{\epsilon^2\delta}$, for Personalized PageRank value $\pi(s^*,v)\ge\delta$, we have*

$$|\hat{\pi}(s^*,v)-\pi(s^*,v)|\le\epsilon\cdot\pi(s^*,v)\tag{6}$$

holds with probability at least $1-\frac{1}{n}$.

By Lemma 5, setting δ to the PageRank lower bound $\frac{\alpha}{n}$ and $\epsilon=\frac{1}{\log^d n}$ gives Lemma 4. □

Since Algorithm 3 gives at least the same accuracy guarantee as Algorithm $\mathcal{A}'$, we immediately have the following lemma.

Lemma 6. *The approximated PageRank values by Algorithm 3 satisfy the accuracy guarantee given by Equation 1 when $\epsilon=\Theta(\frac{1}{\log^d n})$ for a given constant $d\ge1$.*

4.2. Round Complexity and Bandwidth

In this section we present the analysis for the round complexity and bandwidth.

Round Complexity. The Fractional-Push phase will be finished in $O(\log_{\frac{1}{1-\alpha}}\frac{(2\epsilon/3+2)\log(2n)}{\epsilon^2\alpha})=O(\log_{\frac{1}{1-\alpha}}\frac{2\epsilon/3+2}{\alpha}+2d\log_{\frac{1}{1-\alpha}}\log n+\log_{\frac{1}{1-\alpha}}\log(2n))=O(d\log\log n)$ rounds. For the Radar-Push phase, by Lemma 1 each unit task can be finished in $O(\log\log n)$ rounds. The total number of rounds involved is thus $O(d\log\log n)$.

Bandwidth. We show that the bandwidth of Algorithm 3 is $O(d\log^3 n)$ bits. In the Fractional-Push phase, in each synchronized round (i.e., each loop in Line 5 of Algorithm 3), every pair of nodes only transmits a message that is a float value in $[0,1]$ describing the portion of τ being distributed between the nodes. Suppose we use h bits to encode the float value, then the encoding error is up to $\frac{1}{2^h}$. To explain, we can divide the space of $[0,1]$ into 2^h subspaces of equal size. For any float value that falls within the subspace, we use the left boundary of the subspace to encode the value. Since the subspace is of size $\frac{1}{2^h}$, the rounding error is at most $\frac{1}{2^h}$.

In each round of the Fractional-Push Phase, any node u receives $d(u)$ messages, which accumulates to at most $\frac{d(u)}{2^h}$ error, or $O(\frac{d(u)}{2^h})$ error. However, such accumulated error is distributed in the next round to its $d(u)$ neighbors so that only $O(\frac{1}{2^h})$ error passes to the next round. In the meanwhile, in the current round an additional $O(\frac{1}{2^h})$ error in each message is incurred due to rounding. Taking into account that the Fractional-Push phase costs $O(d\log\log n)$ rounds and hence incurs $O(\frac{d\log\log n}{2^h})$ error in each message in the Fractional-Push Phase. As each node receives at most n messages to compute a τ value in each round, the final (maximum) rounding error of a τ value after the Fractional-Push phase is $O(\frac{dn\log\log n}{2^h})$. As we aim to achieve a $\Theta(\frac{1}{\log^d n})$ relative error of the PageRank value approximation, we need a sufficiently large h such that even for the smallest τ , we can still achieve a $\Theta(\frac{1}{\log^d n})$ relative error. For each round, $\tau(u)$ is reduced by a factor of $\frac{1}{d(u)}\cdot(1-\alpha)\geq\frac{1-\alpha}{n}$. Hence, any τ value after the Fractional-Push phase is lower bounded by

$$\frac{1}{n}(\frac{1-\alpha}{n})^{\log_{\frac{1}{1-\alpha}}\frac{(2\epsilon/3+2)\log(2n)}{\epsilon^2\alpha}} \tag{7}$$

It thus requires the following equation in order to achieve an acceptable error (hides a constant factor)

$$\frac{dn\log\log n}{2^h}=\frac{1}{n}(\frac{1-\alpha}{n})^{\log_{\frac{1}{1-\alpha}}\frac{(2\epsilon/3+2)\log(2n)}{\epsilon^2\alpha}}\cdot\frac{1}{\log^d n}$$

which gives us $h=O(d\log n\cdot\log\log n)=O(d\log^2 n)$.

Hence, in the Fractional-Push phase we only need $O(d\log^2 n)$ bits of bandwidth.

In the Radar-Push phase, the bandwidth can be affected by two factors. The first factor is the bandwidth incurred by stitching the short walks in Radar-Push steps. By Lemma 1, each unit task costs $O(\log^2 n)$ bits of bandwidth. Since there are $O(2d\log_{\frac{1}{1-\alpha}}n)$ unit tasks processed in parallel in Algorithm 3 (see Line 15), the total bandwidth of stitching short walks is bounded by $O(2d\log_{\frac{1}{1-\alpha}}n\cdot\log^2 n)=O(d\log^3 n)$ bits. The second factor is the bandwidth incurred by rounding the float value $\tau(s)\cdot\frac{(1-\alpha)^L\cdot\alpha)}{d(s)\cdot\lceil(1-\alpha)^L\cdot\alpha\rceil}$. To guarantee a $\Theta(\frac{1}{\log^d n})$ relative error for any $\pi(u)$, we need a $\Theta(\frac{1}{\log^d n})$ relative error for $\tau(s)\cdot\frac{(1-\alpha)^L\cdot\alpha)}{d(s)\cdot\lceil(1-\alpha)^L\cdot\alpha\rceil}$. By previous analysis, we have

$$\tau\geq\frac{1}{n}(\frac{1-\alpha}{n})^{\log_{\frac{1}{1-\alpha}}\frac{(2\epsilon/3+2)\log(2n)}{\epsilon^2\alpha}}$$

$$\frac{1}{d(s)}\geq\frac{1}{n},\text{ and }\frac{(1-\alpha)^L\cdot\alpha}{\lceil(1-\alpha)^L\cdot\alpha\rceil}\geq(1-\alpha)^L\cdot\alpha$$

Hence, for an acceptable rounding error, we need

$$\frac{1}{2^h}=\frac{1}{n}(\frac{1-\alpha}{n})^{\log_{\frac{1}{1-\alpha}}\frac{(2\epsilon/3+2)\log(2n)}{\epsilon^2\alpha}}\cdot\frac{1}{n}\cdot(1-\alpha)^L\cdot\alpha\cdot\frac{1}{\log^d n}$$

where $L=2d\log_{\frac{1}{1-\alpha}}n$. This gives us $h=O(d\log n\cdot\log\log n)$. Hence, the overall bandwidth is $O(d\log^3 n)$ bits, and we summarize all our results by the following lemma.

Lemma 7. *Given an undirected and connected simple graph $G(V,E)$ of n nodes, and a constant $d\geq 1$, and also given an error bound $\epsilon=\Theta(\frac{1}{\log^d n})$, Algorithm 3 can be implemented in the bandwidth-generalized congested clique model to approximate PageRank in $O(d\log\log n)$ rounds with a bandwidth of $O(d\log^3 n)$ bits. For any node $v\in V$, the approximated PageRank, $\hat{\pi}(v)$, satisfies $|\hat{\pi}(v)-\pi(v)|\leq\epsilon\cdot\pi(v)$ with probability at least $1-\frac{1}{n}$.*

5. Conclusion

We present a new distributed PageRank algorithm that can be implemented in a bandwidth-generalized congested clique model. Our algorithm strategically combines the Fractional-Push operation and the Radar-Push algorithm. We give a detailed analysis on the algorithm, and show that it significantly improves the state-of-the-art algorithm in terms of the bandwidth, while achieving asymptotically the same round complexity.

A. More Details for Algorithm 1

We first note that the lower bound of PageRank value of any node is at least $\frac{\alpha}{n}$ (because there is a probability $\frac{1}{n}$ that a walk starts at each node and then with a probability α the walk ends at that node as well). The unit-tasks that have been cut off (by dropping the range $L\in(\log_{\frac{1}{1-\alpha}}(\frac{1}{\epsilon^*}\cdot\frac{n}{\alpha}),\infty)$ contribute to $\pi(u)$ at most $\frac{\alpha\cdot(1-\alpha)}{n}\cdot\epsilon^*$ absolute error, which is at most $\epsilon^*=\frac{\epsilon}{2}$ relative error. Meanwhile, conducting $\frac{6\log(2n)}{\epsilon^2\alpha}$ random walks guarantees a relative error of $\epsilon^*=\frac{\epsilon}{2}$ as well. Overall the relative error is at most ϵ.

B. More Details for Radar-Push

In the original Radar-Push paper (Luo, 2019), the relative error is considered as a constant. In this paper, we discuss a more general case where the relative error is about $\Theta(\frac{1}{\log^d n})$. We show that in this more general case the round complexity of Radar-Push is about $O(\log\log n + \log d) = O(\log\log n)$. To explain, we first consider that the lower bound of PageRank value is $\frac{\alpha}{n}$. After we do $O(\log_{1-\alpha}(\frac{\alpha}{n} \cdot \frac{1}{\log^d n})) = O(\log n + d \log\log n)$ rounds of α-decay process, only $\frac{\alpha}{n} \cdot \frac{1}{\log^d n}$ portion of the walks are remained, which contribute at most a relative estimation error of $\frac{\frac{\alpha}{n} \cdot \frac{1}{\log^d n}}{\frac{\alpha}{n}} = \Theta(\frac{1}{\log^d n})$. Hence, it is sufficient to only consider walks of length $O(\log n + d \log\log n)$ in Radar-Push to achieve a relative error of $\Theta(\frac{1}{\log^d n})$. By Multi-Phase Radar-Push introduced in (Luo, 2019), we get the round complexity to be $O(\log(\log n + d \log\log n)) = O(\log\log n + \log d)$.

References

Ahmadi, B., Kersting, K., and Sanner, S. Multi-evidence lifted message passing, with application to pagerank and the kalman filter. In *IJCAI*, 2011.

Andersen, R., Chung, F., and Lang, K. Local graph partitioning using pagerank vectors. In *FOCS*, pp. 475–486, 2006.

Beame, P., Koutris, P., and Suciu, D. Communication steps for parallel query processing. *JACM*, 64(6):1–58, 2017.

Berkhin, P. Bookmark-coloring algorithm for personalized pagerank computing. *Internet Mathematics*, 3(1):41–62, 2006.

Das Sarma, A., Molla, A. R., Pandurangan, G., and Upfal, E. Fast distributed pagerank computation. *Theoretical Computer Science*, (561):113–121, 2015.

Fang, Y., Cheng, R., Luo, S., and Hu, J. Effective community search for large attributed graphs. *PVLDB*, 9(12): 1233–1244, 2016.

Florescu, C. and Caragea, C. A position-biased pagerank algorithm for keyphrase extraction. In *AAAI*, 2017.

Fujiwara, Y., Nakatsuji, M., Shiokawa, H., Mishima, T., and Onizuka, M. Fast and exact top-k algorithm for pagerank. In *AAAI*, 2013.

Ghaffari, M. and Parter, M. Mst in log-star rounds of congested clique. In *PODC*, pp. 19–28, 2016.

Guo, T., Cao, X., Cong, G., Lu, J., and Lin, X. Distributed algorithms on exact personalized pagerank. In *SIGMOD*, pp. 479–494, 2017.

Hegeman, J. W. and Pemmaraju, S. V. Lessons from the congested clique applied to mapreduce. *Theoretical Computer Science*, 608:268–281, 2015.

Hegeman, J. W., Pandurangan, G., Pemmaraju, S. V., Sardeshmukh, V. B., and Scquizzato, M. Toward optimal bounds in the congested clique: Graph connectivity and MST. In *PODC*, pp. 91–100, 2015.

Jurdziński, T. and Nowicki, K. Mst in o (1) rounds of congested clique. In *SODA*, pp. 2620–2632, 2018.

Klauck, H., Nanongkai, D., Pandurangan, G., and Robinson, P. Distributed computation of large-scale graph problems. In *SODA*, pp. 391–410, 2014.

Lenzen, C. Optimal deterministic routing and sorting on the congested clique. In *PODC*, pp. 42–50, 2013.

Luo, S. Distributed pagerank computation: An improved theoretical study. In *AAAI*, volume 33, pp. 4496–4503, 2019.

Luo, S., Kao, B., Li, G., Hu, J., Cheng, R., and Zheng, Y. TOAIN: a throughput optimizing adaptive index for answering dynamic k nn queries on road networks. *PVLDB*, 11(5):594–606, 2018.

Luo, S., Cheng, R., Kao, B., Xiao, X., Zhou, S., and Hu, J. ROAM: A fundamental routing query on road networks with efficiency. *TKDE*, 2019a.

Luo, S., Xiao, X., Lin, W., and Kao, B. BATON: Batch one-hop personalized pageranks with efficiency and accuracy. *TKDE*, 2019b.

Luo, S., Xiao, X., Lin, W., and Kao, B. Efficient batch one-hop personalized pageranks. In *ICDE*, pp. 1562–1565, 2019c.

Malewicz, G., Austern, M. H., Bik, A. J., Dehnert, J. C., Horn, I., Leiser, N., and Czajkowski, G. Pregel: a system for large-scale graph processing. In *SIGMOD*, pp. 135–146, 2010.

Neumann, M., Ahmadi, B., and Kersting, K. Markov logic sets: Towards lifted information retrieval using pagerank and label propagation. In *AAAI*, 2011.

Page, L., Brin, S., Motwani, R., and Winograd, T. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab, 1999.

Ponzetto, S. P. and Strube, M. Deriving a large scale taxonomy from wikipedia. In *AAAI*, volume 7, pp. 1440–1445, 2007.

Wang, S., Yang, R., Xiao, X., Wei, Z., and Yang, Y. Fora: simple and effective approximate single-source personalized pagerank. In *SIGKDD*, pp. 505–514, 2017.

Zhu, Y., Ye, S., and Li, X. Distributed pagerank computation based on iterative aggregation-disaggregation methods. In *CIKM*, pp. 578–585, 2005.