An Advanced Parallel PageRank Algorithm

Qi Zhang 1, Zhengan Yao 2, Jun Liang 3, and Zanbo Zhang 4

1 School of Systems Science and Engineering, Sun Yat-sen University, Guangzhou 510275, Guangdong, People’s Republic of China
2 Department of Mathematics, Sun Yat-sen University, Guangzhou 510275, Guangdong, People’s Republic of China
3 School of Software, South China Normal University, Foshan 528225, China
4 School of Statistics and Mathematics, Guangdong University of Finance and Economics, Guangzhou, 510320, China

Corresponding author: Jun Liang, School of Software, South China Normal University, Foshan 528225, China.

ABSTRACT Initially used to rank web pages, PageRank has now been applied in many fields. In general case, there are plenty of special vertices such as dangling vertices and unreferenced vertices in the graph. Existing PageRank algorithms usually consider them as ‘bad’ vertices since they may take troubles. However, in this paper, we propose a parallel PageRank algorithm which can take advantage of these special vertices. For this end, we firstly interpret PageRank from the information transmitting perspective and give a constructive definition of PageRank. Then, based on the information transmitting interpretation, a parallel PageRank algorithm which we call the Information Transmitting Algorithm (ITA) is proposed. We prove that the dangling vertices can increase ITA’s convergence rate and the unreferenced vertices and weak unreferenced vertices can decrease ITA’s calculations. Compared with the MONTE CARLO method, ITA has lower bandwidth requirement. Compared with the power method, ITA has higher convergence rate and generates less calculations. Finally, experimental results on four data sets demonstrate that ITA is 1.5-4 times faster than the power method and converges more uniformly.

INDEX TERMS PageRank, information transmitting, parallel

I. INTRODUCTION

S.Brin and L.Page [1], [2] proposed PageRank when dealing with the problem of ranking web pages retrieved by Google. PageRank measures the importance of web pages based on network’s topology rather than contents. A web page has higher importance if it is linked by more web pages or the web pages link to it are more important. Nowadays, the application of PageRank is far more beyond internet [3] as we can find it in many fields such as social network analysis, chemistry, molecular biology, sports [4], humanities and even social sciences [5].

PageRank was defined recursively in some work [1], [2]. This definition shows that one vertex’s PageRank value is uniquely and directly determined by its neighbors. A big disadvantage is that only graph without unreferenced vertices and dangling vertices has PageRank, thus some preprocessing such as delete dangling vertices or adding artificial links is necessary. Graphs in real world always contain plenty of unreferenced vertices and dangling vertices. For large-scale graph, preprocessing is inefficient.

Parallel is a significant method to speed algorithm. However, there are some challenges when parallelizing existing PageRank algorithms: algorithms based on the power method only partially support parallelizing since the dependency between different iterations cannot be eliminated without extra costs; although algorithms based on the MONTE CARLO method support parallelizing perfectly, acceptable result depends on initialization which may require more memory space and larger bandwidth.

In fact, PageRank is global property rather than local property. The power method implies that one vertex’s PageRank reflects the information of all vertices which can reach it according to the topology. Considering PageRank as the information held by vertices, we can deem the iteration of power method as the process of information transmitting. If we know the information each vertex has transmitted to others, PageRank can be easily obtained. Motivated by this, we propose an advanced parallel PageRank algorithm in this paper. The contributions are as follows:

- We give a constructive definition of PageRank from the information transmitting perspective;
- We propose ITA which can take advantage of the special vertices without any preprocessing;
- We prove that the dangling vertices can increase ITA’s convergence rate and the unreferenced vertices and weak unreferenced vertices can decrease ITA’s calculations;
- We do experiments on four data sets and the results demonstrate that ITA is 1.5-4 times faster than the power method and converges more uniformly.

The remains of this paper are as follows: in section 2, we survey PageRank and the existing algorithms; in section...
3, we interpret PageRank from the information transmitting perspective and give a constructive definition of PageRank; in section 4, we propose the Information Transmitting Algorithm and give a theoretical analysis; some numerical experiments are proceeded in section 5; we summarize this paper in section 6.

II. PAGERANK AND RELATED WORKS

Given directed graph $G(V,E)$ where $V = \{v_1, v_2, \cdots, v_n\}$, $E = \{(v_i, v_j), i,j = 1,2,\cdots, n\}$ and $|E| = m$. Let $A = (a_{ij})_{n \times n}$ denote the adjacency matrix where $a_{ij} = \begin{cases} 1, & \text{if } (v_j, v_i) \in E, \\ 0, & \text{else} \end{cases}$. Let $P = (p_{ij})_{n \times n}$ denote the probability transition matrix where $p_{ij} = \begin{cases} a_{ij} / \sum_{j=1}^{n} a_{ij}, & \text{if } \sum_{j=1}^{n} a_{ij} \neq 0, \\ 0, & \text{else} \end{cases}$. Let $d = (d_1, d_2, \ldots, d_n)^T$ where $d_i$ is 1 if $v_i$ is dangling vertex and 0 else. Let $P_\gamma = P + \gamma d d^T$ and $P'' = c P_\gamma + (1 - c) e e^T$. Denote by $\pi$ the PageRank vector, then one of PageRank’s definitions is as follows:

$$\text{Find } \pi \text{ satisfying }$$

$$\pi = P'' \pi, \pi_i \in (0,1), \sum_{i=1}^{n} \pi_i = 1. \quad (1)$$

PageRank problem is completely determined by $P$, the damping factor $\gamma$ and the personalized vector $\rho$. Thus we can abbreviate PageRank problem as $PR(P,\gamma,\rho)$.

That $PR(P,\gamma,\rho)$ can be interpreted from three different perspectives including geometry, algebra and stochastic process.

- From the geometry perspective, we can interpret Formula (1) as finding the eigenvector corresponding to eigenvalue 1, then $P''$ is linear transformation on linear space $\mathbb{R}^n$.

- From the algebra perspective, by expanding Formula (1), that $PR(P,\gamma,v)$ is equivalent to solving linear system

$$\left(I - \gamma P'\right)\pi = (1 - \gamma)p. \quad (2)$$

- From the stochastic perspective, $\pi$ is equivalent to the stationary distributing of a Markov chain $\{X(n)\}$ whose state space is $V$ and probability transition matrix is $P''$, thus $\pi_i$ is the probability of vertex $v_i$ be visited by a random walk starting at any vertex and walking according to $P''$.

There have been many PageRank algorithms proposed from these three different perspectives. Computing PageRank by iteration $\pi(k + 1) = P'' \pi(k)$ is called the power method from the geometric or algebraic perspective. The power method is simple and stable. There have been a lot of improvements based on the power method. Kamvar [8] focused on the second largest eigenvalue and proposed the extrapolation method by making full use of historical iteration information. Kamvar [9] proposed Block Page Rank and the block method by considering the network’s block structure. Gang [10] proposed the POWER-ARNOLDI algorithm, which took the projection of the exact solution of the linear system (2) on the Krylov subspace as the approximate solution. Some efforts were made to parallelize the power method. Migallon [11] parallelized the power method by making each thread calculating the specified vertices. However, this algorithm was not completely parallel since the $(t+1)$th iteration started only when the $t$th iteration finished. Eedi [12] parallelized the power method by implementing atomic operations in a non-blocking manner. Though this algorithm seemed to eliminate the dependency of different iterations, the convergence rate decreased in fact.

From the stochastic process perspective, the MONTE CARLO method [13] simulates random walks on the directed graph and utilizes $N_i / \sum_i N_i$ approximating $\pi_i$, where $N_i$ is the times of visiting $v_i$. The MONTE CARLO method completely supports parallelising. The convergence depends on the amount of initial random walks, walk length and the forward speed. Lai [14] started with the MCMC problem and then obtained the lower bound of the walk length under the preset accuracy. Sarma [15] introduced the results of the distributed random walk problem [16] into PageRank and accelerated forward speed by stitching short paths to generate long paths. Luo [17], [18] went a step further, recursively stitched long paths had stitched by short paths to generate longer paths, which got an exponential acceleration compared with Sarma’s linear acceleration.

Some scholars proposed distributed PageRank algorithms for large-scale graphs. Sankaralingam [19] proposed the distributed power method on P2P network. This algorithm directly sent the PageRank value of the source vertices to the target vertices thus required larger communication bandwidth. Zhu [20] proposed the aggregation-disaggregation PageRank algorithm, which was distributed and whose bandwidth requirement decreased since there were fewer links between blocks. Stergiou [21] predefined the communication protocol so the message was simplified that decreasing the bandwidth requirement more. Ishii [22] introduced randomization into the PageRank algorithm, which only processed few randomly selected vertices per iteration, thus decreased the bandwidth requirement significantly. There proposed many distributed PageRank algorithms by combining specific algorithms with randomization [23]–[27].

In addition, Zhibo [28]–[30] reduced the scale of the graphs which had the DAG structure by linear transformations. With the development of hardware, some scholars also tried computing PageRank on GPU [31], [32].

There still are shortcomings although the existing PageRank algorithms have their own advantages.

- Some preprocessing such as deleting the dangling ver-
Any improvement based on the power method such as randomizing can not eliminate the dependency between different iterations without extra costs;

- The MONTE CARLO method requires large memory space and communication bandwidth, the accuracy of the result is affected by initialization.

### III. INTERPRETATION FROM INFORMATION TRANSMITTING PERSPECTIVE

Considering Formula (2), since \( I - cP' \) is reversible, we could can that

\[
\pi = (1 - c)(I - cP')^{-1}p. \tag{3}
\]

Since \( \rho(cP') < 1 \), we can obtain that

\[
\pi = (1 - c) \sum_{i=0}^{\infty} (cP')^i p. \tag{4}
\]

Denote by \( S_i(r) \) the set of vertices that belong to \( V \) and are able to reach \( v_i \) after \( r \) steps according to \( P' = (p'_{ij})_{n \times n} \), by \( P'_{k*} \) the \( i \)-th row vector of \( P' \), by \( P'_c \) the \( j \)-th column vector of \( P' \) and by \( G'(V', E') \) the directed graph corresponding to \( P' \). Given vertex \( v_i \), we can have that

\[
\pi_i = (1 - c) p_i + \sum_{s_i(r)} (1 - c) cP'_{s_i}(P')^{-1} + \sum_{s_i(1)} (1 - c) cP'_{s_i}(P')^{-1} + \ldots + (1 - c) cP'_{s_i}(P')^{-1} + \sum_{s_i(r)} (1 - c) cP'_{s_i}(P')^{-1} + \ldots . \tag{5}
\]

Let \( \pi_{ij}(r) = (1 - c) c^{r} P'_{s_i}(P')^{-1} - 2 P'_{s_j} p_j \), then \( \pi_{ij}(r) \) satisfies:

- (a) \( \pi_{ij}(r) \geq 0 \).
- (b) \( \lim_{r \to \infty} \pi_{ij}(r) = 0 \).
- (c) \( \{ \sum_{r=0}^{s} \pi_{ij}(r) \} \) increases monotonically with respect to \( s_i \).
- (d) \( \{ \sum_{j=1}^{n} \sum_{r=0}^{s} \pi_{ij}(r) \} \) increases monotonically with respect to \( s \).

and that

\[
\pi_i = \lim_{s \to \infty} \sum_{j=1}^{n} \sum_{r=0}^{s} \pi_{ij}(r) = \sum_{j=1}^{n} \lim_{s \to \infty} \sum_{r=0}^{s} \pi_{ij}(r) = \sum_{j=1}^{n} \pi_{ij}. \tag{6}
\]

From Formula (4), we can interpret \( PR(P, c, p) \) as the distribution of the amount of information on \( G(V, E) \) after sufficient transmitting according to \( P'' \), where the initial distribution is \( p \). Denote by \( W_{ij} = \bigcup_{r=0}^{\infty} w_{ij}(r) \) the set of the whole paths from \( v_j \) to \( v_i \), where \( w_{ij}(r) \) is the set of the whole paths from \( v_j \) to \( v_i \) with length \( r \), by \( p_{ij}(r) = v_j v_{k_{r-1}} v_{k_{r-2}} \cdots v_{k_1} v_i \in w_{ij}(r) \) one path from \( v_j \) to \( v_i \) with length \( r \), we can decompose \( \pi_i \) as follows:

- The amount of information which \( v_i \) gets from \( v_j \) through \( W_{ij} \) is \( \pi_{ij} \);
- The amount of information which \( v_i \) gets from \( v_j \) through \( w_{ij}(r) \) is \( \pi_{ij}(r) \);
- The amount of information which \( v_i \) gets from \( v_j \) through \( p_{ij}(r) \) is \( (1 - c) c^r p_{ik_1} p_{k_1 k_2} \cdots p_{k_{r-1} k_r} p_{kj} \).

Formula (4) can be regarded as a constructive definition of PageRank, based on which any directed graph, even if one contains isolated vertices and self-loops, has unique PageRank. Moreover, Formula (6) means that the PageRank value of \( v_i \) is determined by the whole vertices which can reach it according to \( P' \). The initial distribution of the amount of information is corresponding to the personalized vector \( p \) and is generally taken as \( \frac{1}{\theta} \). Next, we commence to design algorithms based on the constructive definition.

### IV. INFORMATION TRANSMITTING ALGORITHM

Given \( p_{ij}(r) = v_j v_{k_{r-1}} v_{k_{r-2}} \cdots v_{k_1} v_i \in w_{ij}(r) \), let \( d(p_{ij}(r)) = \deg(v_j) \deg(v_{k_{r-1}}) \cdots \deg(v_{k_1}) \deg(v_{k_0}) \). We call \( r \) the path length and \( d(p_{ij}(r)) \) the path accumulation degree. Then the first algorithm is given as the Basic Information Transmitting Algorithm.

Algorithm 1 is completely parallel, however, there exist many obvious shortcomings. Firstly, Algorithm 1 may generate plenty of messages and as a result that large memory space and communication bandwidth are required; secondly, storing messages by queue is costly. Denote by \( \theta \) the average degree of \( G(V, E) \), by \( r(t) \) the average path length at time \( t \) and by \( N(t) \) the total amount of messages, then we can obtain that \( N(t) = O(n \theta^r(t)) \). The memory space requirement can be reduced by decreasing the total amount of messages and simplifying a single message.

Given \( G(V, E) \), transforming \( P \) into \( P' \) means linking each dangling vertex to all vertices and thus increases the average degree \( \theta \). In fact, it is unnecessary based on the constructive definition. Expanding Formula (2), we can have
Algorithm 1 Basic Information Transmitting Algorithm

**Input:** The upper bound of the path length $R$ and the upper bound of the path accumulation degree $D$.

**Output:** PageRank $\pi$.

1. Each vertex $v_i$ maintains a variable $\pi_i$ and a message queue $QUE$, the message is constructed by $(id, r, d)$, where $id$ is the identifier of $v_i$, $r$ is the path length and $d$ is the path accumulation degree.
2. Initially each vertex $v_i$ sets $\pi_i = 0$ and pushes $msg = \langle id(v_i), 0, 1 \rangle$ into $QUE$.
3. while $QUE.size() > 0$ do ▶ [Each $v_i$ does in parallel]
   4. $tmp = QUE.front();$
   5. $QUE.pop();$
   6. if $tmp.r < R$ and $tmp.d < D$ then
      7. $\pi_i = \pi_i + \frac{(1-c) \times tmp.r}{\sum_{i}^{} tmp.r};$
   8. $\pi_i = \min[\pi_i + \frac{1}{3};$
   9. $\pi_i = \pi_i + \frac{1}{3};$
   10. for $u \in \text{DEST}(v_i)$ do ▶ $\text{DEST}(v_i)$ is the set of target vertices of $v_i$ according to $P$
      11. $u.QUE.push(tmp);$
   12. end for
   13. end if
   14. end while
   15. Calculate $\pi$ following $\pi_i = \frac{\pi_i}{\sum_{i=1}^{N} \pi_i}$.

\[
(I-cP)\pi = (cd^TP + (1-c))\pi.
\]

That $(I-cP)\pi = (cd^TP + (1-c))\pi$ is reversible, we can have that

\[
\pi = \gamma (I-cP)^{-1} \pi,
\]

where $\gamma = (cd^TP + (1-c))$. Since $1 = e^T \pi = e^T (I-cP)^{-1} \pi$, we can have that $\gamma = \frac{1}{e^T (I-cP)^{-1} \pi}$. Since $i_1 \pi = \gamma e_1^T (I-cP)^{-1} \pi$, where $e_1$ is an $n$-dimensional column vector whose $i_1$th element is 1 and 0 others, we can have that $\pi_i = \frac{e_1^T (I-cP)^{-1} \pi}{e_1^T (I-cP)^{-1} \pi}$. Let $Q = \{q_{ij}\}_{i=1}^{n} \times n = (I-cP)^{-1}$, we can have that $\pi_i = \frac{\sum_{j=1}^{n} q_{ij} p_j}{\sum_{j=1}^{n} \sum_{j=1}^{n} q_{ij} p_j}$. Expanding $(I-cP)^{-1}$, we can have that

\[
q_{ij} = I_{ij} + cp_{ij} + c^2 \sum_{k_1=1}^{n} p_{ik_1} p_{k_1 j} + \cdots + c^d \sum_{k_d=1}^{n} \cdots \sum_{k_{d-1}=1}^{n} p_{ik_d} \cdots p_{k_{d-1} j}.
\]

That $\sum_{j=1}^{n} q_{ij} p_j$ is the amount of information of $v_i$ contributed by $v_j$ after sufficient transmitting according to $P$, i.e., $\pi_{ij} = \sum_{j=1}^{n} q_{ij} p_j$. More important, transmitting according to $P$ means that the transmitting can be terminated at dangling vertices and as a result the amount of messages decreases. Given message $\langle id, r, d \rangle$ held by $v_i$, $v_i$ does the following two:

(a) $\pi_i = \pi_i + (1-c) \frac{c^dp_{id}}{d}$;
(b) Sending $\frac{c}{deg(v_i)} \frac{c^dp_{id}}{d}$ to all the target vertices of $v_i$.

We can deem $\frac{c^dp_{id}}{d}$ as the amount of information carried by message $\langle id, r, d \rangle$, then $c$ portion is accumulated into PageRank value and $1-c$ portion is transmitted to the dest vertices. Only scalar $\frac{c^dp_{id}}{d}$ is transmitted, so that messages and queues are not necessary. The second algorithm is given as the Improved Information Transmitting Algorithm.

Algorithm 2 Improved Information Transmitting Algorithm

**Input:** The lower bound of the amount of information $\xi$.

**Output:** PageRank $\pi$.

1. Each vertex $v_i$ maintains a data structure $(\pi_i, h_i)$, where $\pi_i$ is the amount of information that $v_i$ has received and $h_i$ is the amount of information that $v_i$ holds currently.
2. Initially each vertex $v_i$ sets $\pi_i = 0$, $h_i = 1$.
3. while $h_i > \xi$ do ▶ [Each $v_i$ does in parallel]
   4. $\pi_i = \pi_i + (1-c)h_i$;
   5. for $u \in \text{DEST}(v_i)$ do ▶ $\text{DEST}(v_i)$ is the set of target vertices of $v_i$ according to $P$
      6. $h_u = h_u + \frac{c \pi_i}{deg(v_i)}$;
   7. end for
   8. $h_i = 0$;
   9. end while
   10. Calculate $\pi$ following $\pi_i = \frac{\pi_i}{\sum_{i=1}^{N} \pi_i}$.

Algorithm 2 requires $O(n)$ memory space and $O(1)$ bytes bandwidth and thus decreases the memory space and bandwidth requirement compared with algorithm 1, however, there still exists shortcomings. Firstly, assigning processing thread for each vertex is unrealistic; secondly, dealing the dangling vertices is unnecessary.

$\{\pi_{ij}(v)\}$ satisfies the commutative and associative law, thus the processing order, i.e., sending the amount of information to the target vertices at different times, has no effect on the final results. Since $\pi_i = \frac{\pi_{ij}(v)}{\sum_{i=1}^{N} \pi_i}$, we can still obtain $\pi$ by taking $\pi_i = h_i$ but not $\pi_i = (1-c)h_i$. This property means that dealing the dangling vertices is unnecessary. We can generate $K$ calculation threads and assign non-dangling vertices to them. Then the third algorithm is given as the Advanced Information Transmitting Algorithm.

Updating $h_u$ must be atomic since $u$ may have many source vertices. Appropriate $K$ according to the computing environment makes full use of the computing resource. Un-referenced vertices and weak unreferenced vertices on the DAG structure will exit computing as soon as converged, i.e., algorithm 3 is self-adaptive. Self-adaptation lowers the time complexity and will be detailed in the next section. We call algorithm 3 as the Information Transmitting Algorithm in the following and abbreviate it as ITA.

A. ITA AND FORWARD PUSH

Although Forward Push [23], which is one of the state-of-the-art Personalized PageRank (PPR) algorithms, is similar to ITA in form, there are differences between them:

- Forward Push is sequential while ITA is parallel;
- Forward Push deals all the vertices while ITA deals only non-dangling vertices;

IEEE OPEN JOURNAL OF THE INDUSTRIAL ELECTRONICS SOCIETY. VOLUME XX, 20XX
Algorithm 3 Advanced Information Transmitting Algorithm

Input: The thread number \( K \), the lower bound of the amount of information \( \xi \).
Output: PageRank \( \pi \).

1. Each vertex \( v_i \) maintains a data structure \( (\pi_i, h_i) \), where \( \pi_i \) is the amount of information that \( v_i \) has received and \( h_i \) is the amount of information that \( v_i \) holds currently.
2. Assign all non-dangling vertices to the \( K \) calculation threads, denote by \( S_i \) the set of vertices belonging to thread \( j \), denote by \( ATO \) a global bool variable.
3. Initially each vertex sets \( \pi_i = 0, h_i = 1 \) and \( ATO = true \).
4. Invoke Management; \( \triangleright \) [The \( K \) calculation threads and the management thread do in parallel]
5. for \( j = 1..K \) do
6. \quad Invoke Calculation(j);
7. end for
8. Calculate \( \pi \) following \( \pi_i = \frac{\pi}{\sum_{i=1}^{n} \pi_i} \) while all threads terminated.
9. 10: function \text{Calculation}(j)
11: while \( ATO \) do
12: \quad for \( v_i \in S_i \) do
13: \quad \quad if \( h_i > \xi \) then
14: \quad \quad \quad \quad \pi_i = \pi_i + h_i;
15: \quad \quad for \( u \in \text{DEST}(v_i) \) do \( \triangleright \) [\( \text{DEST}(v_i) \) is the set of target vertices of \( v_i \) according to \( P' \)]
16: \quad \quad \quad \quad \quad h_u = h_u + \frac{\pi_i}{\deg(v_i)};
17: \quad \quad end do
18: \quad \quad h_i = 0;
19: \quad end if
20: \quad end for
21: \quad end while
22: end function
24: 25: function \text{Management}
26: while \( ATO \) do
27: \quad Count the amount \( CNT \) of non-dangling vertices satisfying that \( h_i > \xi \); 
28: \quad if \( \text{CNT} = 0 \) then
29: \quad \quad ATO = False;
30: \quad end if
31: end if
32: end while
33: end function

Forward Push treats \( \pi \) as approximate PageRank while ITA treats \( \frac{\pi}{\sum_{i=1}^{n} \pi_i} \) as approximate PageRank.

V. ITA ANALYSIS

In the following, we will analyze the ITA including time complexity and accuracy and then compare ITA with the MONTE CARLO method and the power method separately. Before all of these, we discuss the distribution of the amount of information on \( G(V, E) \) during ITA running.

For convenience, we deem the calculation threads iterate through \( V \) as one iteration. Denote by \( t \) the \( t_{th} \) iteration of ITA, by \( T \) the number of iterations for ITA getting converged, by \( \pi^I(t) \) the amount of information that has been accumulated into the PageRank value and by \( \pi^R(t) \) the amount of information remaining to be transmitted. Then we can have that

\[
\pi^I(t) + \pi^R(t) = n, \quad (9)
\]

where \( \pi^I(0) = 0 \), \( \pi^R(0) = n \), \( \pi^I(t) \) increases monotonically with respect to \( t \) and \( \pi^R(t) \) decreases monotonically with respect to \( t \).

With ITA running, unreferenced vertices and weak unreferenced vertices exit. Denote by \( V_U(t) \) the set of vertices that have exited at time \( t \) and by \( V_D \) the set of dangling vertices. Let \( \alpha(t) = \frac{|V - V_U(t) - V_D|}{|V_U(t)|} \), we can have that \( \frac{\pi^R(t)}{\pi^R(t-1)} = \alpha(t-1) \) and that

\[
\pi^R(T) = \sum_{v_t \in V - V_D} h_t(t) = n \prod_{t=1}^{T} \alpha(t-1). \quad (10)
\]

That \( \prod_{t=1}^{T} \alpha(t-1) \) can be viewed as ITA’s convergence rate.

Let \( \alpha = \max\{\alpha(t)\} \). ITA gets converged when each non-dangling vertex \( v_i \) satisfies that \( h_i < \xi \), so the prerequisite of ITA gets converged is that

\[
n \prod_{t=1}^{T} \alpha(t-1) < |V - V_U(T) - V_D| \xi. \quad (11)
\]

A. TIME COMPLEXITY OF ITA

We estimate the time complexity from the iteration rounds and operation amounts aspect.

1) Iteration rounds

It is difficult to obtain the iteration rounds for ITA getting converged, however, we can estimate it based on Formula (11). In general case, we assume that \( \pi^R(t) \) evenly scatters in \( V - V_U(t) \) but rather concentrates on several vertices, then ITA gets converged when \( T \) satisfies that

\[
n \prod_{t=1}^{T} \alpha(t-1) < \frac{|V - V_U(T) - V_D| \xi}{n}. \quad (12)
\]

So we can have that

\[
T > \log_{\alpha c} \xi + \log_{\alpha c} \left[ \frac{|V - V_U(T) - V_D|}{n} \right]. \quad (13)
\]

Let \( \lambda = \alpha c \), we can have that

\[
T = O(\log_\lambda \xi). \quad (14)
\]

---

IEEE OPEN JOURNAL OF THE INDUSTRIAL ELECTRONICS SOCIETY. VOLUME XX, 20XX
Formula (14) shows that dangling vertices can decrease λ and thus increase ITA’s convergence rate.

2) Operation amounts
Denote by \( m(t) \) the operation amount at the \( t \)-th iteration and \( M(T) \) the total operation amount for ITA getting converged. Algorithm 3 shows that only vertices belonging to \( V - V_U \) (transmit amount of information to their dest vertices at the \( t \)-th iteration round, so we can have that
\[
m(t) = \sum_{v_i \in V - V_U(t)} \text{deg}(v_i)
\]
and thus
\[
M(T) = \sum_{t=0}^{T} \sum_{v_i \in V - V_U(t)} \text{deg}(v_i). \tag{15}
\]
Since \( \sum_{v_i \in V} \text{deg}(v_i) = m \), we can have that \( M(T) < mT \) and in general case that
\[
M(T) = o(m \log \lambda \xi). \tag{16}
\]
Formula (15) shows that unreferenced vertices and weak unreferenced vertices can decrease ITA’s calculations.

B. ACCURACY
In general case, for the convenience of discussion, we assume that \( \pi^T(t) = n\lambda^t \) and that \( T = \log \lambda \xi \). Thus from Formula (9), we can have that
\[
|\pi^T(t + \epsilon) - \pi^T(t)| = n|\lambda^T - \lambda^{T+\epsilon}| = (1 - \lambda^\epsilon)n\lambda^T. \tag{17}
\]
When \( t = 1 \), we can have that
\[
\text{res}(\xi) = \frac{|\pi^T + 1 - \pi^T(T)|}{n} = (1 - \lambda)\xi. \tag{18}
\]
When \( t \to \infty \), we can have that
\[
\text{err}(\xi) = \lim_{t \to \infty} \frac{|\pi^T(t + \epsilon) - \pi^T(t)|}{\pi^T(t + \epsilon)} = \lim_{t \to \infty} \frac{\lambda^T - \lambda^{T+\epsilon}}{1 - \lambda^\epsilon}. \tag{19}
\]
ITA’s accuracy is directly determined by \( \xi \), the smaller \( \xi \) is, the higher the accuracy is.

C. THE COMPARISON BETWEEN ITA AND MONTE CARLO METHOD
Avrachenkov [13] proposed the MC complete path stopping at dangling nodes algorithm (which we called MC algorithm) with detailed convergence analysis. The MC algorithm and ITA are closely related.

- Both the MC algorithm and ITA are based on the probability transition matrix \( P \), i.e., the original directed graph without preprocessing;
- For ITA, the amount of information held by \( v_i \) is evenly distributed to the target vertices; for MC algorithm, random walks held by \( v_i \) randomly select the target vertices; they come to the same result from a statistical perspective when the number of random walks is large enough;

- The MC algorithm takes the frequency of visits as PageRank while the ITA takes the proportion of the amount of information as PageRank, i.e., PageRank values are calculated in the same form.

ITA is equivalent to the MC algorithm when the number of random walks trends to infinite. MC algorithm can be regarded as a discrete version of ITA and ITA can be regarded as a fractional version of MC algorithm where \( \pi_t \) converges to \( \pi \), with probability 1.

Table 1 shows the comparison between the ITA and the MONTE CARLO methods on time complexity, bandwidth and memory space. Theoretically, in general case, the ITA has lower time complexity, bandwidth requirement and memory space requirement.

| Algorithm | Time complexity | Bandwidth | Memory | Notes |
|-----------|----------------|-----------|--------|-------|
| IPRA [15] | \( O(\sqrt{\log n}) \) | \( O(\log n^3) \) | \( O(n \log n^3) \) | undirected graph |
| BPRA [15] | \( O(\log n^3) \) | \( O(\log n^3) \) | \( O(n \log n^3) \) | directed graph |
| FPRP [18] | \( O(\log n^3) \) | \( O(\log n^3) \) | \( O(n \log n^3) \) | directed graph |
| ITA | \( O(\tau) \) | \( O(\tau) \) | \( O(\tau) \) | directed graph |

### D. THE COMPARISON BETWEEN ITA AND POWER METHOD
We compare ITA with the power method in terms of the convergence rate and the operation amounts. Denote by \( T_P \) the iteration rounds for the power method getting converged, by \( T_I \) the iteration rounds for ITA getting converged, by \( M_P \) the operation amounts for the power method getting converged and by \( M_I \) the operation amounts for ITA getting converged.

- The convergence rate of the power method is \( c^t \) while ITA’s convergence rate is \( \prod_{i=1}^{t}\alpha(i-1) < c \). In general case, ITA requires less iteration rounds than the power method, i.e., \( T_P > T_I \);
- The power method generates \( O(2m + n) \) operations per iteration, so \( M_P = (2m + n)T_P \). In general case, \( M_P > mT_I > M_I \), i.e., ITA generates less operation than the power method.

**Theorem 1:** In general case, given the accuracy \( \xi \), the number of iterations required for ITA getting converged is \( O(\log \lambda \xi) \); ITA’s calculation is \( o(m \log \lambda \xi) \); ITA’s bandwidth requirement is \( O(1) \) bytes; and ITA’s memory space requirement is \( O(n) \) bytes.

### VI. EXPERIMENTS
#### A. EXPERIMENTAL SETTING
In this section, we experimentally demonstrate ITA’s performance. First of all, we show the convergence of ITA; then we compare ITA with other PageRank algorithms; finally,
we show ITA’s uniform convergence. Let \( RES = ||\pi(k) - \pi(k - 1)||_2 \), where \( \pi(k) \) is the result of the \( k_{th} \) iteration for the power method and the result with \( \xi = 10^{-k} \) for ITA. Let \( ERR = \max_{v_i \in V} \{ \frac{||\pi_i - \pi_i'||}{\pi_i} \} \), where \( \pi_i \) is the true PageRank value of \( v_i \). Let \( c = 0.85 \) and take the result of 210\(_{th}\) iterations of the power method as the true value. Denote by \( T \) the consumption of CPU time. ITA processes with 8 threads and multi-threaded power method processes with 32 threads. All of the three algorithms are implemented by C++ multi-threading technology. Table 2 shows the hardware and software utilized by these numerical experiments. Table 3 shows all four data sets where \( n \) is the amount of vertices, \( m \) is the amount of edges, \( nd \) is the amount of dangling vertices and \( \text{deg} = \frac{m}{n} \).

### TABLE 2. Hardware and software

| CPU | Intel(R) Core(TM) i7-10510U CPU 1.80GHz 2.30GHz |
|-----|-----------------------------------------------|
| Memory | 16G |
| OS | Windows 10 64bits |
| Database | Oracle Database 19c Enterprise Edition Release 19.00.03.0 |
| C++ compiler | Visual Studio 2019 |

### TABLE 3. Data sets

| Data sets | \( n \) | \( m \) | \( nd \) | \( \text{deg} \) |
|-----------|------|------|------|------|
| web-Stanford\(^1\) | 281903 | 2512497 | 172 | 8.21 |
| Stanford-Berkeley | 685446 | 7583376 | 68062 | 12.32 |
| web-Google | 875713 | 5105039 | 130259 | 6.90 |
| in-2004 | 1382870 | 16917053 | 282268 | 15.37 |

### B. THE CONVERGENCE OF ITA

The relationship between \( RES, T \) and \( \xi \) on four data sets is shown in Figure 1.

1. The blue lines show that \( RES \) is positively related with \( \xi \) and exhibits an obvious linear relationship, which is consistent with Formula (18).
2. The red lines show that \( T \) is negatively related with \( \xi \) and exhibits an exponential relationship, which is consistent with Formula (14).
3. The red solid lines and red dotted lines show that atomic operation implemented by mutex is better than which implemented by spin lock, since spin lock executes testing all the time.
4. The blue lines show that ITA’s accuracy seems have boundary. We suppose that it is not a theoretical shortcoming but caused by data type’s insufficiency on precision. Since the number of significant digit of C++ double type is 15, \( h_i \) which is less than \( 10^{-15} \) can not change any \( \pi_i \), thus \( RES \) stays when \( \xi < 10^{-15} \).

### C. THE COMPARISON BETWEEN ITA AND POWER METHOD

We compare ITA with the single-threaded power method(SPI) \(^1\) and \(^2\) and the multi-threaded power method(MPI) \(^3\). The relationship between \( RES \) and \( T \) is shown in Figure 2. The relationship between \( ERR \) and \( T \) is shown in Figures 3.

1. The blue lines and green lines in Figure 2 and Figure 3 show that MPI is faster than SPI at any accuracy although they have the same convergence rate. MPI processes matrix-vector multiplication in parallel and utilizes CPU more sufficiently. For example, on Stanford-Berkeley the CPU utilization is 30%-40% for SPI, while 60% for MPI.
2. The blue lines and red lines in Figure 2 and Figure 3 show that ITA is faster than SPI at any accuracy. Table 4 shows that ITA is 1.5-4 times faster than SPI when \( ERR < 0.001 \). This result is guaranteed by ITA’s higher convergence rate and parallelism.
3. Theoretically, ITA should be faster than MPI at any accuracy, but the green lines and red lines in Figure 2 and Figure 3 show the contrary results on Stanford-Berkeley and in-2004. Next, we will study that how different number of threads affect ITA and MPI.

---

\(^1\)https://sparse.tamu.edu/SNAP/web-Stanford

FIGURE 1. \( \xi \) versus \( RES \) and \( T \)

FIGURE 2. \( T \) versus \( RES \)
Taking Stanford-Berkeley for example, we compare MPI with ITA on \( T \) and CPU utilization at different thread number, where MPI takes the 200th iteration and ITA takes \( \xi = 10^{-14} \). Figure 3 shows the relationships between \( T \), CPU utilization and thread number \( K \).

![Figure 3. \( T \) versus ERR](image)

The blue solid line shows that MPI’s CPU utilization increases but not above 70% with thread number’s increasing. The red solid line shows that MPI’s CPU time consumption decreases but not under 10 seconds with thread number’s increasing. Only \( P\pi(k) \) can parallel in MPI, thus both CPU utilization and CPU time consumption have limits to which the result are close.

The blue dotted line shows that ITA’s CPU utilization increases with thread number’s increasing, but the CPU has been exhausted with 8 calculation threads. The red dotted line shows that ITA’s CPU time consumption decreases with thread number’s increasing when there are less than 8 threads. Since the lack of computing resource, the result fails to demonstrate ITA’s real performance.



| Data sets         | SPI  | MPI  | ITA  |
|-------------------|------|------|------|
| web-Stanford      | 4.819| 3.045| 1.38 |
| Stanford-Berkeley | 5.532| 3.674| 4.151|
| web-Google        | 11.378| 6.157| 3.419|
| in-2004           | 13.444| 8.876| 7.345|

**TABLE 4.** The time consumption when \( ERR < 0.001 \)

Denote by \( M \) the total operation amount, by \( \beta \) the clock ticks per operation, by \( k \) the number of paralleling threads and by \( s \) the CPU utilization per thread. If \( ks < 1 \), i.e., there’s enough computing resource, we can have that

\[
T = \left( \frac{M \delta}{k} + M(1 - \delta) \right) \beta, \quad (20)
\]

where \( \delta \) is the proportion of operations that are able to parallel. Let \( T_{\text{mpi}} = \left( \frac{M_{\text{mpi}} \delta_{\text{mpi}}}{k_{\text{mpi}}} + M_{\text{mpi}}(1 - \delta_{\text{mpi}}) \right) \beta_{\text{mpi}} \), \( T_{\text{ita}} = \left( \frac{M_{\text{ita}} \delta_{\text{ita}}}{k_{\text{ita}}} + M_{\text{ita}}(1 - \delta_{\text{ita}}) \right) \beta_{\text{ita}} \). Since \( \delta_{\text{mpi}} = \frac{2m}{nk}, \delta_{\text{ita}} = 1 \), we can have that

\[
\frac{T_{\text{ita}}}{T_{\text{mpi}}} = \frac{\beta_{\text{ita}} M_{\text{ita}}}{\beta_{\text{mpi}} M_{\text{mpi}}} \frac{1}{k_{\text{ita}}} \frac{\delta_{\text{mpi}}}{\delta_{\text{ita}}} + 1 - \delta_{\text{mpi}}. \quad (21)
\]

Assuming that the computing resource is enough for MPI, when \( k_{\text{mpi}} \) is large enough, we can have that

\[
\frac{T_{\text{ita}}}{T_{\text{mpi}}} = \frac{\beta_{\text{ita}} M_{\text{ita}}}{\beta_{\text{mpi}} M_{\text{mpi}}} \frac{1}{k_{\text{ita}}} \frac{2m}{nk} \frac{m+n}{nk_{\text{ita}}}.
\]

ITA is faster than MPI when \( \frac{T_{\text{ita}}}{T_{\text{mpi}}} < 1 \). Formula (22) shows that \( \frac{T_{\text{ita}}}{T_{\text{mpi}}} \) is determined by \( \beta_{\text{ita}}/\beta_{\text{mpi}}, M_{\text{ita}}/M_{\text{mpi}} \) and \( m/n, m/nk, \) we discuss them separately.

- Several clock ticks are enough for MPI to finish an addition while ITA takes dozens or even hundreds of clock ticks to complete an atomic addition, so we can have that \( \beta_{\text{ita}} > \beta_{\text{mpi}} \);
- In general case, dangling vertices, unrefrenced vertices and weak unrefrenced vertices decrease the total operation amount of ITA, so we can have that \( M_{\text{ita}}/M_{\text{mpi}} < 1 \);
- Given \( G(V, E), T_{\text{ita}}/T_{\text{mpi}} \) is mainly determined by \( m/n, m/nk \), that \( T_{\text{ita}}/T_{\text{mpi}} < 1 \) can be true if \( k \) is large enough, i.e., there’s sufficient computing resource.

However, Figure 4 shows that the computing resource exhaust with 8 threads. For graph with large \( m/n \), such as Stanford-Berkeley and in-2004, ITA is slower than the power method if there’s no sufficient computing resource.

![Figure 4. Threadnum versus \( T \) and CPU occupation](image)

**D. UNIFORM CONVERGENCE**

Figure 5 shows the relationship between \( ERR \) and \( RES \) on four data sets. For all four data sets, ITA has smaller \( ERR \) when \( RES \) of ITA is equal to \( RES \) of MPI. It means that ITA converges more uniformly than the power method.

**VII. CONCLUSION AND FUTURE WORK**

In this paper, we propose ITA which is parallel and can take advantage of special vertices. The dangling vertices can increase ITA’s convergence rate and the unrefrenced vertices and weak unrefrenced vertices can decrease ITA’s calculations. Both theoretical analysis and experimental results shows that ITA performs better on the graph with
large portion of special vertices. However, there still are some shortcomings: ITA is inefficient on undirected graph; ITA's accuracy is limited by the data type; Atomic addition implemented by mutex costs too much clock ticks. Therefore, there still are room for improving ITA in the future. Having obtained the most fine-grained decomposition of PageRank, we can continue discussing PageRank on dynamic graph.

REFERENCES
[1] S. Brin and L. Page, “Reprint of: The anatomy of a large-scale hypertextual web search engine,” Computer networks, vol. 56, no. 18, pp. 3825–3833, 2012.
[2] L. Page, S. Brin, R. Motwani, and T. Winograd, “The pagerank citation ranking: Bringing order to the web,” Stanford InfoLab, Tech. Rep., 1999.
[3] D. F. Gleich, “Pagerank beyond the web,” Siam Review, vol. 57, no. 3, pp. 321–363, 2015.
[4] S. Brown, “A pagerank model for player performance assessment in basketball, soccer and hockey,” arXiv preprint arXiv:1704.00583, 2017.
[5] Ying, Hou, Zhou, Dou, Wang, and Shao, “The changes of central cultural cities based on the analysis of the agglomeration of literati’s footprints in tang and song dynasties,” Journal of Geo-information Science, vol. 22, no. 5, pp. 945–953, 2020.
[6] S. Kamvar, T. Haveliwala, and G. Golub, “Adaptive methods for the computation of pagerank,” Linear Algebra and its Applications, vol. 386, pp. 51–65, 2004.
[7] T. Haveliwala, S. Kamvar, D. Klein, C. Manning, and G. Golub, “Computing pagerank using power extrapolation,” Stanford, Tech. Rep., 2003.
[8] 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, pp. 261–270, 2003.
[9] S. Kamvar, T. Haveliwala, C. Manning, and G. Golub, “Exploiting the block structure of the web for computing pagerank,” Stanford, Tech. Rep., 2003.
[10] G. Wu and Y. Wei, “A power-arnoldi algorithm for computing pagerank,” Numerical Linear Algebra with Applications, vol. 14, no. 7, pp. 521–546, 2007.
[11] H. Migallón, V. Migallón, and J. Penadés, “Parallel two-stage algorithms for solving the pagerank problem,” Advances in Engineering Software, vol. 125, pp. 188–199, 2018.
[12] H. Eedi, S. Peri, N. Ranabothu, and R. Utkoor, “An efficient practical non-blocking pagerank algorithm for large scale graphs,” in 2021 29th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP), pp. 35–43. IEEE, 2021.
[13] K. Avrachenkov, N. Litvak, D. Nemirovsky, and N. Osipova, “Monte carlo methods in pagerank computation: When one iteration is sufficient,” SIAM Journal on Numerical Analysis, vol. 45, no. 2, pp. 890–904, 2007.