Abstract—In this paper we present a new method that can accelerate the computation of the eigenvector of the transition matrix associated to the PageRank algorithm. Our method, called D-Iteration, is based on the decomposition of the matrix-vector product that can be seen as a fluid diffusion model and is potentially adapted to asynchronous implementation. We also make a comparison with OPIC (Online Page Importance Computation) which also relies on a diffusion approach to estimate importance of web pages. We show through experimentation using real datasets on which criteria and how much our method can improve the computation efficiency.

I. INTRODUCTION

PageRank is a link analysis algorithm that has been initially introduced in [1] and used by the Google Internet search engine. It assigns a numerical value to each element of a hyper-linked set of nodes, such as the World Wide Web. The algorithm may be applied to any collection of entities (nodes) that are linked through directional relationships. The numerical value assigned to each node is called PageRank and is associated to an eigenvector problem of which we are interested in the computation issue. The complexity of computing this vector of a matrix increases rapidly with the dimension of the vector space. Efficient and accurate methods to compute eigenvalues and eigenvectors of arbitrary matrices are in general a difficult problem. In the particular case of PageRank equation, several specific solutions were proposed and analysed [2, 3] including power method [1] with adaptation [4] or extrapolation [5, 6], or adaptive online method [7], etc.

In this paper we present some theoretical results of D-Iteration (DI): mathematical definition, convergence to the fixed point, measurement of the distance to the limit and solution for dynamic graph update issues. This work points out how DI improves the OPIC algorithm [7]. The two methods are based on a diffusion approach and consist in dealing with two vectors: fluid vector in DI (or cash vector in OPIC) and history vector in both of them. The comparison focuses on three aspects: the convergent fixed point, stopping condition and the distance to the limit. As we will see later in the paper, given a target error (or precision) history vector in both of them. The comparison focuses on three aspects: the convergent fixed point, stopping condition and the distance to the limit. As we will see later in the paper, given a target error (or precision) history vector of DI and the distance to the limit. As we will see later in the paper, given a target error (or precision) history vector of DI, it is also possible to manage the distance of how far the history vector is to the fixed point based on the residual fluid remaining in the system.

II. PAGE RANK EQUATION

The term PageRank is well-known today as the name of an algorithm used to assign numerical values to nodes in a directed graph, particularly web graph, where nodes can be web pages and edges are hyper-links among them. This value represents how important a node is. By definition, a node is considered important if it is pointed to by other important nodes. This statement requires to look for a fixed point vector by solving iteratively the equation:

\[ PR(j) = \frac{1 - d}{N} + d \sum_{i \rightarrow j} \frac{PR(i)}{outdeg(i)} \]  

(1)

where \( PR(j) \) is the PageRank score of page \( j \); \( i \) is incoming neighbor of \( j \) and \( d < 1 \) (\( d = 0.85 \) [8]) is the damping factor. This ranking is used as one of the metrics of Google search engine to determine how to order the pages returned by a web search query. As a matter of fact, the rapid explosion of Google in recent years affirms an undeniable success of this algorithm.

III. D-ITERATION

A. Diffusion model

In this part, we talk about the intuition of DI. Our approach consists in a succession of diffusion operations (fluid of a node passing to its children following its outgoing links).

Now we take a general look of how collection and diffusion approach work. Given a directed three-nodes graph in Figure 1 collection methods make use of incoming links of nodes as in Figure 2 (or rows of the corresponding transition matrix) whereas diffusion method exploits outgoing links as in Figure 3 (or columns of the matrix). If the iteration is based on vector level update (such as Jacobi or Power iteration), the collection or diffusion approaches become equivalent (full cycle operations on all nodes). Somehow, these two types of operations can be seen as dual operations, but with different consequences. With our diffusion approach, we will show

Dohy Hong \&†, The Dang Huynh \&†, Fabien Mathieu \&‡

†Alcatel-Lucent Bell Labs France ‡LINCS

{dohy.hong, the_dang.huynh, fabien.mathieu}@alcatel-lucent.com
We consider a non-negative matrix \( P \) of size \( N \times N \) such that each column sums up to one. In particular, we can associate such a matrix to a Markov chain of \( N \) states, where \( P \) is the transition matrix and \( P_{ij} \) is the transition probability from state \( j \) to state \( i \). Each column of \( P \) sums up to one means that in the Markov chain, each state \( j \) has a positive probability to jump to at least one state \( i \) (if such a condition is not true, we complete the matrix \( P \) by replacing the zero columns with \( \mathbf{1}/N \) (\( \mathbf{1} \) is the column vector with all components equal to one) or by the personalization vector \( V \) in the general case) to get \( \overline{P} \). This corresponds to the dangling nodes completion [2] and we will see below that such an adaptation is not required in our approach.

In this paper, we consider the iteration of equations of the form:

\[
x_{n+1} = P x_n \tag{2}
\]

where \( P \) is a matrix of size \( N \times N \) which can be explicitly decomposed as:

\[
P = dP + (1 - d)V \overline{1} \tag{3}
\]

where \( V \) is a normalized vector of size \( N \). Note that all entries of \( x \) sum up to one, so we have:

\[
x_{n+1} = dP x_n + (1 - d)V \tag{4}
\]

We assume that the stationary probability of \( P \) is defined by the vector \( x \) (thanks to the PageRank matrix \( P \) associated to a Markov chain of \( N \) states). We have:

\[
x = dP x + (1 - d)V \tag{5}
\]

and as a consequence:

\[
x_{\infty} = (1 - d) \sum_{k=0}^{\infty} d^k P^k V = (1 - d)(I - dP)^{-1} V \tag{6}
\]

where \( I \) is the identity matrix.

### C. Main equations

In the following, we assume that a deterministic or a random diffusion sequence \( I = \{i_1, i_2, \ldots, i_k, \ldots\} \) with \( i_k \in \{1, \ldots, N\} \) is given. We only require that the number of occurrences of each value of \( i_k \) in \( I \) is infinite. This condition ensures the fairness of the algorithm where all nodes are chosen infinitely often to diffuse its fluid.

With DI we have to deal with two variable vectors at the same time: fluid vector \( F \) and history vector \( H \). The fluid vector \( F \) associated to \( I \) is defined as follows:

\[
F_0 = (1 - d)V \tag{7}
\]

\[
F_n = dP J_n F_{n-1} + \sum_{k \neq i_n} J_k F_{n-1} \tag{8}
\]

\[
= (I - J_n + dP J_{i_n})F_{n-1} \tag{9}
\]

where \( J_k \) is the matrix with all entries equal to zero except the \( k^{th} \) diagonal term \( (J_k)_{kk} = 1 \). We then define the history vector \( H \):

\[
H_0 = \vec{0}
\]

\[
H_n = \sum_{k=1}^{n} J_k F_{k-1}. \tag{10}
\]

By definition, \( H_n \) is an increasing function (all components are positive). DI is presented in Algorithm 1.

**Algorithm 1** D-Iteration algorithm: \( x = dPx + (1 - d)V \)

1: \( H[i] = 0; \) \( \triangleright \) Initializing history vector
2: \( F[i] = (1 - d)V[i]; \) \( \triangleright \) Initializing diffusion vector
3: \( k = 1; \)
4: while \( \lvert F \rvert > \text{Target Error} \) do
5: \( \quad \) Choose some node \( i_k; \)
6: \( \quad \) sent = \( F[i_k]; \)
7: \( \quad \) \( F[i_k] = 0; \)
8: \( \quad \) \( H[i_k] += \text{sent}; \)
9: \( \quad \) for all child node \( j \) of \( i_k \) do
10: \( \quad \quad \) \( F[j] += \text{sent \times } dP_{jik}; \)
11: \( \quad \) end for
12: \( \quad k++; \)
13: end while
Theorem 1. We have the equality:

\[ H_n + F_n = F_0 + dPH_n. \]  

(10)

Proof: The proof is straightforward by induction: assuming the equation (10) true for \( n \):

\[
\begin{align*}
H_{n+1} + F_{n+1} &= H_n + J_{i,n+1}F_n + \sum_{k \neq i,n+1} J_kF_n + dPJ_{i,n+1}F_n \\
&= H_n + F_n + dPJ_{i,n+1}F_n \\
&= F_0 + dPH_n + dPJ_{i,n+1}F_n \\
&= F_0 + dPH_{n+1}.
\end{align*}
\]

D. Convergence

Theorem 2. For any fair sequence \( I \), the history vector \( H_n \) is convergent to the unique vector \( x \) such that \( x = dPx + (1-d)V \):

\[
\lim_{n \to \infty} H_n = (1-d)(I-dP)^{-1}V
\]

Proof: From the equation (10) we have:

\[
H_n = (I-dP)^{-1}(F_0 - F_n)
\]

\[
= (I-dP)^{-1}F_0 + \sum_{k=0}^{\infty} d^k P^k F_n
\]

Because \( F_0 = (1-d)V \) and \( F_n \) tends to zero (thanks to the fair sequence \( I \) and damping factor \( d < 1 \)), so:

\[ H_n \leq (1-d)(I-dP)^{-1}V \]

By construction \( H_n \) is increasing per entry, it is thus convergent to the claimed limit which is exactly the unique vector \( x \) by (6).

Theorem 3. We have the equality:

\[
x_\infty - H_n = \sum_{k=0}^{\infty} d^k P^k F_n = (I-dP)^{-1}F_n
\]

(11)

and as a direct consequence, we have:

\[
|x_\infty - H_n| = \frac{|F_n|}{1-d}
\]

(12)

where \( | \cdot | \) is the \( L_1 \) norm.

Proof: We have:

\[
x_\infty = dPx_\infty + F_0
\]

Using the equation (10), we have:

\[
x_\infty - H_n = dP(x_\infty - H_n) + F_n
\]

\[
= \sum_{k=0}^{\infty} d^k P^k F_n
\]

By iteration, we get (11). Note that \( P \) is a left stochastic matrix (square matrix of non negative real numbers and each column sum up to one), thus:

\[
|P^k F_n| = |F_n|
\]

and as consequence:

\[
|x_\infty - H_n| = \sum_{k=0}^{\infty} d^k |F_n| = \frac{|F_n|}{1-d}
\]

Remark 1. This distance to the limit is only correct if \( P \) is left stochastic (e.g. thanks to dangling node completion) such that the condition \( |P^k F_n| = |F_n| \) is satisfied. The general distance to the limit is discussed in [11,12].

As the distance to the limit depends on the remaining fluid \( |F_n| \), one can recognize that the convergence speed of the algorithm relates explicitly to the diffusion sequence. Such a good strategy therefore would consist in making the fluid vanish as quickly as possible. This disappearance is caused by two main parameters: damping factor \( d \) and dangling nodes (which have minor impact and concerns a normalization factor [11,12]). Assume that a node contains a fluid \( f \) at the time it is chosen to diffuse. Due to the factor \( d < 1 \), an amount equal to \( df \) passes to its children and concurrently, an amount \((1-d)f\) leaves the system. Apparently, the best strategy is to apply diffusion on the node having \( f_{\max} \) but the searching cost in this case is expensive. We propose a simple strategy called DI-argmax as follows: at \( i^{th} \) iteration, we only diffuse nodes having \( f \) greater than the average fluid in the whole graph of \( (i-1)^{th} \) iteration (or equal to \( (1-d) \times (1/N) \)) in the first iteration). The fairness of this strategy is apparent since \( |F_n| \) is a decreasing function so that a certain node containing fluid \( f \) will be eventually diffused at the time \( f > |F_{n-1}|/N \).

By using one additional variable to keep track of the total fluid amount of the previous iteration, this strategy can help detecting with a negligible cost if a node is worth a diffusion in the current iteration.

E. Distance to the limit

In the PageRank context, we may have zero-columns in \( P \) (if we don’t do the P completion operation). Indeed if those columns (corresponding to dangling nodes) are completed by \( 1/N \), any iteration scheme would do useless computations. If we work on the matrix \( P \) without completion, the limit we obtain needs to be renormalized (by a constant multiplication for diffusion approach or by constant addition for power iteration).

To take into account of this effect precisely, we count the total amount of fluid that has left the system when a diffusion was applied on a dangling node (for the sake of simplicity, we consider the case \( V = 1/N \)). We call this quantity \( e_n \) (at step \( n \) of the DI). This quantity should have been put in the system by adding \( e_n \times d/N \) on each node, which means that the initial fluid of each node should have been \((1-d+de_n)/N\) instead of \((1-d)/N\). But then the fluid \( de_n/N \) would have produced after \( n \) steps \(((de_n)^2/(1-d))/N\) that disappears again by dangling nodes, etc. With respect to the initial condition \(|F_0|\),
we have:
\[ |F_0| = (1 - d) \cdot \frac{d_{e_n}}{1 - d} \cdot \frac{d_{e_n}}{1 - d} + \frac{d_{e_n}}{1 - d} \cdot \frac{d_{e_n}}{1 - d} \cdot \frac{d_{e_n}}{1 - d} \cdot \frac{d_{e_n}}{1 - d} \cdots \]
\[ \frac{d_{e_n}}{1 - d} \cdot \frac{d_{e_n}}{1 - d} \cdot \frac{d_{e_n}}{1 - d} \cdot \frac{d_{e_n}}{1 - d} \cdots \]
Applying the argument recursively, the correction that is required on the residual fluid \(|F_n|\) is to replace the initial condition \(|F_0| = (1 - d)\) by \(|F_0'|\):
\[ |F_0'| = (1 - d) + d_{e_n} \cdot \sum_{n=0}^{\infty} \left( \frac{d_{e_n}}{1 - d} \right)^n \]
\[ = (1 - d) + d_{e_n} \cdot \frac{1}{1 - d - d_{e_n}} \]
As \(|F_0'|/|F_0| = (1 - d)/(1 - d - d_{e_n})\), \(H_n\) needs to be renormalized (multiplication) by \((1 - d)/(1 - d - d_{e_n})\) so that the exact \(L_1\) distance \(|x_\infty - H_n|\) is equal to:
\[ \frac{|x_\infty - H_n|}{1 - d - d_{e_n}} = \frac{|F_n|}{1 - d - d_{e_n}}. \]
In the DI approach, we update \(e_n\) by:
\[ e_n = (F_n)_{i_n} \]
if \(i_n\) is a dangling node.

**F. Update equation**

The fact that \(H_n\) converges to \(x_\infty\) for any arbitrary choice of the sequence \(I\) can be also exploited to compute more efficiently the new eigenvector in case of graph modification \((P\) is modified to \(P')\).

**Theorem 4.** Assume the initial graph associated to \((P, H_\infty)\) is modified to the updated graph represented by \((P', H_\infty')\), then we have:
\[ H_\infty' - H_\infty = (1 - dP')\cdot 1\cdot d(P' - P)H_\infty. \]

**Proof:** Knowing that \(F_0' = \hat{0}\), from (10) we have:
\[ (1 - dP')H_\infty' = F_0 \]
and thus:
\[ (1 - dP')(H_\infty' - H_\infty) = F_0 - (1 - dP')H_\infty = F_0 - (1 - dP - dP' + dP)H_\infty = F_0 - (1 - dP)H_\infty + d(P' - P)H_\infty = d(P' - P)H_\infty. \]

Using D-Iteration with the initial condition \(F_0 = d(P' - P)H_\infty\) and \(H_0 = \hat{0}\), we have the convergence:
\[ \lim_{n \to \infty} H_n = H_\infty' - H_\infty. \]

Now assume that the vector \(H_n\) has been computed up to the iteration \(n_0\) and at that time, we are interested in computing the limit associated to \(P'\) (for instance, because the web graph has been modified/updated).

Then very naturally, we can apply our diffusion method with \(P'\), but with the modified initial condition \(F_0' = F_{n_0} + d(P' - P)H_{n_0}\) for which we have:
\[ H_n' + F_n' = F_0' + dP'H_n'. \]

We have the following intuitive results:

**Theorem 5.** \(H_{n_0} + H_\infty' (H_\infty' is the limit of the equation (13)) is the solution of the equation:
\[ X = dP'X + F_0. \]

**Proof:** The limit of (13) satisfies:
\[ H_\infty' = F_0' + dP'H_\infty' \]
\[ = F_{n_0} + d(P' - P)H_{n_0} + dP'H_\infty'. \]

Combining this with \(H_{n_0} + F_{n_0} = F_0 + dPHP_{n_0}\), we have the claimed result:
\[ H_\infty' = (F_0 + dPHP_{n_0} - H_{n_0}) + dP'H_{n_0} - dPHP_{n_0} + dP'H_\infty' \]
\[ = (dP'(H_\infty' + H_{n_0}) + F_0) - H_{n_0}. \]

The above result implies that one can continue the diffusion process when \(P\) is regularly updated by just injecting in the system a fluid quantity equal to \(d(P' - P)H_{n_0}\) and then applying the new matrix \(P'\). If a distributed computation is to be used, we just need to synchronize the time from which \(P'\) is applied.

**Remark 2.** The existing iterative methods (such as Jacobi, Gauss-Seidel iteration, Power Iteration, ...) can naturally adapt the iteration to the modification of \(P\) because they are in general initial condition independent (for any starting vector, the iterative scheme converges to the unique limit). This is not the case of DI and why the above result is important.

**IV. OPIC Algorithm**

The OPIC proposed in [7] is a method used to compute the importance score of nodes in a graph. This algorithm is also based on the diffusion approach which is different from the methodology of the classical PageRank: nodes diffuse their importance score through outgoing links (in OPIC) instead of collecting the score by incoming links (in PageRank). Given the same PageRank matrix, both methods converge to the same PageRank vector. In OPIC, each page \(i\) is initially given an amount of cash and each time a page is chosen to be iterated (or crawled), its current cash \(\hat{C}[i]\) is equally distributed to outgoing neighbours and at the same time is credited to the history record \(H[i]\). At a given moment, the importance of a
page $k$ is proportional to the cash flow going through it and is equal to $(C[k] + \mathcal{H}[k])/(G + 1)$ where $G$ is the total cash accumulated. The OPIC is presented in Algorithm

Because the sum of cash moving in the system is always constant and equals to unity (i.e., $\sum_{i=1}^{N} C[i] = 1$), OPIC does not have an explicit stopping condition.

### Algorithm 2 OPIC Algorithm

1. $\mathcal{H}[i] = 0; \quad \triangleright\text{ Initializing history vector}$
2. $C[i] = 1/N; \quad \triangleright\text{ Initializing cash vector}$
3. $G = 0; \quad \triangleright\text{ Cash}$
4. while (true) do
5.   Choose some node $i$;
6.   $\mathcal{H}[i] += C[i]$;
7.   for all child node $j$ of $i$ do
8.     $C[j] += C[i]/\text{out}[i]$;
9.   end for
10. $G += C[i]$;
11. $C[i] = 0$;
12. end while

Assume that $C_t$ and $\mathcal{H}_t$ are the values of vector $C$ and $\mathcal{H}$ at the end of the $t^{th}$ step of the algorithm; and let $X_t$ be defined by:

$$X_t = \frac{\mathcal{H}_t}{|\mathcal{H}_t|}$$

where $| \cdot |$ is the $L_1$ norm. It has been proven that $|X_t| = 1$ and the vector $X_t$ converges to the vector of importance:

$$\lim_{t \to \infty} X_t = X_{\text{importance}}$$

Note that if one applies the PageRank matrix to OPIC (detailed in [V]), the obtained vector $X_{\text{importance}}$ is exactly the PageRank vector. As there is no explicit stopping condition in OPIC, the algorithm can be executed until a desired precision $\epsilon$ is reached:

$$|X_{t+1} - X_t| < \epsilon$$

The only constraint of the selection sequence is the fairness: all nodes must be read infinitely often. However, the selection strategy has an impact on the convergence speed of the algorithm. There are three schemes considered in the original paper:

- Random: nodes are read in a random order.
- Cycle: nodes are read in a fixed order.
- Greedy: the next node with highest cash is read.

It has been empirically shown that given the same number of nodes read, the Greedy strategy converges faster than the Random and Cyclic variant since the error term is inversely proportional to the norm of the history vector $|H_t|$. However, as the cost of looking for the node having highest cash is expensive when the graph grows large, we implement a strategy called OPIC-max (see [V]) which has the flavour of DI-max and also converges more quickly than the other two versions.

### V. Experiments and Comparisons

In this section, we compare DI with OPIC running on real web graph structures. We also put Power Iteration (PI) as a conventional standard for the evaluation.

#### A. Experiment setup

We consider three web graphs available at [9]:

1. **sk-2005**: contains 50 million nodes and approximately 2 billion links. This is a crawl of the .sk (Slovakian) domain with a very large number of seeds.
2. **webbase-2001**: contains 120 million nodes and over 1.1 billion links. This graph was created by the WebBase [10] crawler.
3. **uk-06-07**: 133 million nodes and 5.5 billion links. This crawl has been done for DELIS project [11] and is a one-year snapshot (from May 2006 to May 2007) of .uk domain.

The properties of the three graphs are summarized in Table I with some notations:

- **N**: number of nodes in the graph (i.e. number of columns/rows in matrix);
- **L**: number of links (i.e. number of non-null entries in matrix $P$);
- **max_in**: maximum in-degree of a node (i.e. row with maximum non-null entries in matrix);
- **max_out**: maximum out-degree of a node (i.e. column with maximum non-null entries in matrix);
- **D**: number of dangling nodes (i.e. nodes with no outgoing link, columns with only null entries);
- **O**: number of self-loop nodes.

We define the matrix $P$ of size $N \times N$ as follows:

$$P_{ij} = \begin{cases} 1/\text{outdeg}(j) & \text{if link } j \rightarrow i \text{ exists.} \\ 0 & \text{otherwise.} \end{cases}$$

where $P_{ij}$ is the entry at $i^{th}$ row and $j^{th}$ column of matrix $P$; $\text{outdeg}(j)$ is out-degree of node $j$. With DI, the zero columns of the matrix $P$ does not need to be completed as long as the normalization factor is kept.

We solve the equation using DI:

$$x = dPx + (1 - d)V$$

where $V = [1/N, ..., 1/N]$, $d = 0.85$ and the initial fluid vector $F_0 = [1/N, ..., 1/N]$. Let $F_i[k]$ denote the fluid amount of node $k$ at the $i^{th}$ iteration, we implement two DI variants:

- **DI-cyc**: $k = 1, 2, ..., N$: at $i^{th}$ iteration, diffuse $F_i[k]$ iff $F_i[k] \neq 0$.
- **DI-max**: at $i^{th}$ iteration, diffuse $F_i[k]$ iff $F_i[k] > r$ where $r = \sum_k F_{i-1}[k]/N$.

In case of OPIC, the original version does not take into account of damping factor. In order to have a consistent input with DI, some modifications are made: (a) all zero-columns of the matrix $P$ are replaced by the uniform column vector $[1/N, ..., 1/N]^t$ to have the completed matrix $P$, then...
(b) applying the damping factor $d = 0.85$ to construct matrix $P$ as follows:

$$P = dP + (1 - d)V$$

and (c) modifying the diffusion operation with respect to the matrix $P$ as in Algorithm 3. With these changes, the algorithm still holds its properties (e.g., $\sum_{i=1}^{N} C_t[i] = 1$ at any $t^{th}$ step) and is convergent to an importance vector $X'_{importance}$ which is the same as the PageRank vector produced by DI or PI. We implement two OPIC strategies:

- **OPIC-cyc**: $k = 1, 2, ..., N$: at $i^{th}$ iteration, diffuse $C_i[k]$ iff $C_i[k] \neq 0$.
- **OPIC-argmax**: at $i^{th}$ iteration, diffuse $C_i[k]$ iff $C_i[k] > r = 1/N$.

Remind that the cash amount $C$ is always constant and is equal to 1, in OPIC-argmax we fix the diffusion condition $r = 1/N$ for all iterations.

**Algorithm 3 Modified diffusion operation of OPIC**

1. ...
2. for all child node $j$ of $i$
3. \[ C[j] \leftarrow C[i] \times \frac{P}{j}; \]
4. end for
5. ...

To have a comparison standard, we also put PI which comprises of a succession of matrix vector multiplication:

$$x_{n+1} = Px_n = [dP + (1 - d)V]x_n.$$  

**B. Results**

We now observe the experiment results of sk-2005 in Figure 4a, webbase-2001 in Figure 4b and uk-06-07 in Figure 4c. In three cases, the $y$-axis indicates the precision of the current vector $x_k$ at $k^{th}$ iteration compared to the real PageRank vector $x_\infty$ (the distance to the limit or $|x_\infty - x_k|$). With DI, it is not necessary to know in advance the vector $x_\infty$ (i.e., $|x_\infty - x_k|$ is deduced from $|F_n/(1 - d - de_n)|$) while in OPIC and PI, $x_\infty$ was precomputed using DI with the error $\epsilon = |F_n/(1 - d - de_n)| < 10^{-9}$. The $x$-axis counts the number of rounds required to reach to a certain precision. One round equals to either one full matrix-vector multiplication of PI or diffusion of $N$ nodes in OPIC and DI.

In the original paper, the precision of the OPIC and its variants is evaluated up to the $6^{th}$ round, but the smallest value is approximately equal to $10^{-2}$. For this reason, we decide to stop $y$-axis in our plots at $10^{-2}$. Moreover, thanks to this scale it is easy to compare all algorithms at the $6^{th}$ round.

As one can observe, given the same number of rounds, OPIC-argmax converges more quickly than OPIC-cyc which always stays monotone. On the other hand, if we take into considering the first 6 rounds, OPIC-argmax is a good choice compared to PI and DI-cyc. In all experiments, DI-argmax outperforms its competitors and reaches the precision $10^{-2}$ after several rounds with a gain factor over 2 with respect to the best of the others. This can be intuitively understood because only nodes having sufficiently large fluid

| Dataset name | N   | L   | max_in | max_out | L/N  | D/N  | O/N |
|--------------|-----|-----|--------|---------|------|------|-----|
| sk-2005      | 50,636,154 | 1,949,412,601 | 8,563,308 | 12,870 | 38.50 | 14.0 | 0.23 |
| webbase-2001 | 118,142,155 | 1,019,903,190 | 816,127 | 3,841 | 8.63  | 23.0 | 0.25 |
| uk-06-07     | 133,633,040 | 5,007,679,822 | 6,366,525 | 22,429 | 41.22 | 24.0 | 0.24 |

**TABLE I: Datasets information**

![Fig. 4: Convergence to the stationary PageRank vector](image-url)
are diffused. It then results in a less effort wasted on minor nodes and thus a performance boost.

VI. OPEN ISSUES

Based on the properties of DI, there are some issues interesting to be discussed:

A. Distributed computation

As being explained in the previous sections, our algorithm is capable to adapt to asynchronous computation. Remind that some classical methods are restricted by how to iterate nodes. For instance, Gauss-Seidel updates its vector at element level: it applies right away the vector elements \( x_j^{(k)} \) to compute \( x_i^{(k)} \) for \( j < i \) in the \( k^{th} \) iteration and this procedure impedes the asynchronous deployment. On the contrary, DI has almost no constraint on diffusion sequence except the fairness which is easy to meet. By the way, controlling the fluid spread on several machines in distributed computation is not easy because the issues involves in how graph is partitioned (assigning certain groups of nodes to one or several machines), the way of adapting diffusion strategy and also the distributed database implementation.

B. Optimization problems

The performance of the algorithms is evaluated by the number of iterations or the number of diffusion times. If the target is to optimize the number of elementary operations (i.e., addition, multiplication) done by CPU, one can think about another diffusion strategy considering also the number of outgoing links in the node selection process. The smaller the out-degree a node has, the less operation it does for passing the fluid to its children.

C. Generalizing update equation

We only consider in this study the matrix \( P \) update in terms of links modification. In fact, a complete graph update comprises of node/link additions/removals. This will bring a huge impact on the update strategy, for example how to modify the fluid while having new node arrivals. Obviously, the problem is even more difficult to tackle since we do not know the dynamicity of the graph and its evolution rate.

VII. CONCLUSION

In this paper, we proposed an algorithm to solve PageRank equation based on diffusion approach. We demonstrated some theoretical results concerning the correctness (convergence), the precision measurement and update equation. Our algorithm shows its potential through experiments on real data in comparison with OPIC and PI. For future works, we basically plan to focus on how to adapt and implement DI in a distributed manner. Related problems (graph partitioning,...) of course would be also the subject of future investigations.

REFERENCES

[1] L. Page, S. Brin, R. Motwani, and T. Winograd, in The PageRank Citation Ranking: Bringing Order to the Web., no. 1999-66. Stanford InfoLab, November 1999, Technical Report.
[2] A. N. Langville and C. D. Meyer, “Deeper inside pagerank,” Internet Mathematics, vol. 1, p. 2004, 2004.
[3] M. Bianchini, M. Gori, and F. Scarselli, “Inside pagerank,” ACM Trans. Internet Technol., vol. 5, no. 1, pp. 92–128, Feb. 2005.
[4] S. Kamvar, T. Haveliwala, and G. Golub, “Adaptive methods for the computation of pagerank,” Stanford InfoLab, Technical Report 2003-26, April 2003.
[5] T. Haveliwala, S. Kamvar, D. Klein, C. Manning, and G. Golub, “Computing pagerank using power extrapolation,” Stanford InfoLab, Technical Report 2003-45, 2003.
[6] S. Kamvar, T. Haveliwala, C. Manning, and G. Golub, “Extrapolation methods for accelerating pagerank computations,” in In Proceedings of the Twelfth International World Wide Web Conference. ACM Press, 2003, pp. 261–270.
[7] S. Abiteboul, M. Preda, and G. Cobena, “Adaptive on-line page importance computation,” in Proceedings of the 12th international conference on World Wide Web, ser. WWW ’03. New York, NY, USA: ACM, 2003, pp. 280–290.
[8] S. Brin and L. Page, “The anatomy of a large-scale hypertextual web search engine,” in Proceedings of the seventh international conference on World Wide Web 7, ser. WWW7. Amsterdam, The Netherlands: Elsevier Science Publishers B. V., 1998, pp. 107–117.
[9] Webgraph, “Laboratory of webgraph algorithmic,” [http://webgraph.dii.unimi.it/]
[10] “The stanford webbase project,” [http://www-diglib.stanford.edu/]
[11] I. Bordino, P. Boldi, D. Donato, M. Santini, and S. Vigna, “Temporal evolution of the uk web,” in ICDM Workshops, 2008, pp. 909–918.