MCHITS: Monte Carlo based Method for Hyperlink Induced Topic Search on Networks

Zhaoyan Jin, Dianxi Shi, Quanyuan Wu, and Hua Fan
National Key Laboratory for Parallel and Distributed Processing, School of Computer, National University of Defense Technology, Changsha 410073, China
Email: jinzhaoyan@163.com, dxshi@nudt.edu.cn, wqy.nudt@gmail.com, fh.nudt@gmail.com

Abstract—**Hyperlink Induced Topic Search (HITS) is the most authoritative and most widely used personalized ranking algorithm on networks. The HITS algorithm ranks nodes on networks according to power iteration, and has high complexity of computation. This paper models the HITS algorithm with the Monte Carlo method, and proposes Monte Carlo based algorithms for the HITS computation. Theoretical analysis and experiments show that the Monte Carlo based approximate computing of the HITS ranking reduces computing resources a lot while keeping higher accuracy, and is significantly better than related works.**

Index Terms—**Social Network, Influence Ranking, Random Walk, Monte Carlo Method**

I. Introduction

Node influence on social networks refers to the behavioral change of individuals affected by others in a network. Social influence is an intuitive and well-accepted phenomenon in social networks. The influence of a node on social networks depends on many factors, such as the strength of relationships between nodes in the networks, the network distance between nodes, temporal effects, characteristics of networks and individuals in the network. Influence ranking of nodes on networks, or influence ranking for short, ranks nodes according to their importance on networks. Influence ranking can be used in web search, link prediction or friend recommendation in social networks, and recognition of important authors and papers in author-paper networks. Classical influence ranking algorithms include PageRank [1], HITS [2], SimRank [3, 4], etc.

The PageRank algorithm ranks nodes on a graph offline, and when a request is issued from a user, it returns related results with the PageRank order. Although the offline computation improves the efficiency of the search engine a lot, the long-time computation makes the ranking results lag, and this cannot satisfy users' personalized requests. HITS and its improvements, such as SALSA [5] and ACHITS [6], rank only a small subset of a whole network, and needs less computing resources than PageRank. However, these algorithms are request-dependent, and they must be computed online in order to answer personalized requests of users. As the rapid increase of Internet data and user accesses, the HITS algorithm cannot satisfy users' requests efficiently.

In order to shorten the response time of users' requests and return users with accurate results, this paper models the HITS algorithm with random walk; secondly, approximates the HITS algorithm according to the Monte Carlo method and proposes Monte Carlo based approximation algorithms for HITS; finally, according to the theoretical analysis of these algorithms, we do some experiments to validate the efficiency of the proposed method. The experiments show that the Monte Carlo based approximate method of HITS ranking reduces the computing resources a lot while keeping higher accuracy, and is significantly better than related works.

The rest of the paper is organized as follows: section 2 introduces some background and related work; section 3 describes the MCHITS method and three Monte Carlo based approximate algorithms; theoretical analysis is given in section 4 and experiments are given in section 5; conclusion and future work is in section 6.

II. Related Work

A. Background

Considering a Webpage as a node and the hyperlink between two Webpages as a directed edge, we have a directed graph $G = (V, E)$, among which the V is denoted with the set of Webpages and the E is denoted with the linkages between them, and the numbers of nodes and edges are $|V|$ and $|E|$. The adjacency matrix of the graph G is denoted with M, and the transition possibility matrix of a graph G is denoted with P.

Given an edge (i, j) in E, the node i is the parent of the node j, the node j is the child of the node i, and the weight of (i, j) is denoted with $w_{i,j}$. For the sake of simplifying the presentation of some of the following formulae, we assume that the weights on the outgoing edges of each node sum up to 1.

1) Random Walk and Monte Carlo Method on Graphs

Random walk on a graph is that: choose a source node on the graph; at each random step, go to a candidate node randomly with the probability ε, and jump to a designated node with the probability 1-ε; repeat the above random step until the random walk terminates. Random walk on a graph is the idea that models and analyzes some of the problems on a graph.

Monte Carlo method, also known as statistical simulation method, which is based on probability statistics, solves some problems with (pseudo) random number. On solving the problems on a graph, we can model and analyze them with random walk, and apply Monte Carlo method to approximate the results.

2) PageRank

The PageRank algorithm considers that the influence ranking of a node is related with its authority. The authority of a node is proportional to its parents' authority, and is inverse proportional to the number of its parents' children. The idea of random walk is that: start from the source node *s*; at each random step, go to a child of the current node randomly; repeat the above random step *k* times.

When the number *k* tends to infinite, that is $k \to \infty$, the probability of stability that the random walk stops on each node tends to a constant, and whichever node we choose as the source node, the probability of stability of each node doesn't change. The PageRank algorithm considers that the probability of stability of each node is its authority. According to the authority of each node, we can rank all nodes on a graph. The authority vector $\boldsymbol{\pi}$ of a graph can be computed in the following formula:

$$\pi = P \cdot \pi \tag{1}$$

In order to compute the stable probability of each node, either power iteration of linear algebra [1] or Monte Carlo method [7, 8] can be used. The ides of Monte Carlo based PageRank computation is that: starting from each node $v \in V$, do a number of random walks; at each random step, stop with a probability ε, and jump to a neighbor node randomly with the probability 1-ε. When all random walks are stopped, compute the frequency of each node in all these fingerprints (a fingerprint is the set of nodes in a random walk), and approximate the stable probability of a node with its visited frequency.

3) HITS

The HITS algorithm considers that the influence ranking of a node is decided by two attributes, authority and hub. The authority is decided by citation of the node, and the hub reflects the navigation of a node. The authority vector ***a*** and the hub vector ***h*** can be computed in the following formula:

$$\begin{aligned} \boldsymbol{a} &= M \cdot \boldsymbol{h} \\ \boldsymbol{h} &= M^T \cdot \boldsymbol{a} \end{aligned} \tag{2}$$

The idea of random walk based HITS influence ranking is that [9]: start from a source node *s*; at each random step, if the number of the current step is odd, go to a child randomly, and if else, go to a father randomly; repeat the above random step *k* times. When *k* is odd, and $k \to \infty$, the probability of stability of each node tends to be a constant, and however we choose *s*, the probability of stability doesn't change. The odd probability of stability of a node is its authority. With the same reason, the even stable probability of a node is its hub.

B. Related Work

In this paper, we aim to apply Monte Carlo method for HITS computation, so we review related works about Monte Carlo method for ranking on graphs and the HITS ranking.

1) Monte Carlo Method for Ranking on Graphs

PageRank [1] is originally proposed to rank web pages. While a request is sent by user, the search engine return related results to user with the pre-computed PageRank order. However, the ranking of a web page is always topic sensitive [10], i.e. a web page has different rankings for different topic. If we use power iteration of linear algebra to compute the PageRank ranking, then we need massive resources, and this will take a long time. However, the results, that take a long time to execute, will be out-of –date for the users’ timely requests.

The Monte Carlo method [7, 8] simulates several real random walks, and ranks nodes on a graph according to the frequency of a need in such walks. This provides an efficient and quick approximation method for node ranking. The idea of Monte Carlo method is that: starting from a source node, at each random step, stop the walk with a probability ε, and jump to a neighbor randomly with a probability 1-ε. As the probability of stopping is ε, the length of the walk is distributed geometrically with Geom(ε), and the expectation of length is $1/\varepsilon$. In order to reduce the uncertainty of long fingerprints, Bahmani et al. [12] fix the length at $1/\varepsilon$ for each random walk. Based on the study of Ref. [12], Bahmani et al. [13] design an efficient Monte Carlo method for PageRank computation on MapReduce [14].

2) HITS Ranking

The HITS algorithm considers that the influence ranking attribute of a node is reflected by its authority and hub, and they have mutual reinforcement. Although power iteration can solve this problem, the computation is very large, and thus many new methods appear. The SALSA algorithm [5] transforms the original graph into two graphs, and computes the influence attributes by analyzing their eigenvectors. In order to reduce the online processing time, Gollapudi etc. [15] find the neighbors of each node offline, and use them to answer users’ requests. ACHITS [8] takes advantages of the structure of a graph, and accelerates the converging of HITS by importing two diagonal matrices, and this accelerates the convergence of power iteration a lot.

In order to improve the accuracy of HITS, the Co-HITS [16] algorithm takes the structure and contents of the bipartite graph into consideration while ranking nodes on a graph. Besides the structure of a graph, Li et al. [17] also take the contents of a graph and the weight of edges into consideration. Moreover, Bharat and Henzinger study the reinforcement of authority and hub in HITS in Ref. [18], and Nomura studies the topic drift in HITS in Ref. [19]. The HITS algorithm can not only rank nodes on graphs, but also rank users of social networks [20] and recognize useless linkages on networks [21].

III. MCHITS

This paper substitutes the adjacency matrix *M* in HITS with the transition possibility matrix *P*, proposes a random walk model for HITS influence ranking, and designs three Monte Carlo based approximate algorithms for HITS, short for MCHITS.

A. Definition

The main idea of MCHITS is that: the influence attributes of a node is decided by authority and hub; the authority of a node is proportional to the hub of its fathers and inverse proportional to the number of children of its father; and the hub of a node is proportional to the authority of its children and inverse proportional to the number of fathers of its child. The authority vector $\boldsymbol{a}$ and the hub vector $\boldsymbol{h}$ can be computed in the following formula:

$$\begin{aligned} \boldsymbol{a} &= P \cdot \boldsymbol{h} \\ \boldsymbol{h} &= P^T \cdot \boldsymbol{a} \end{aligned} \tag{3}$$

Figure 1 illustrates the authority/hub results of nodes *a* and *b*. The authority of *a* is decided by *c*, *d* and the numbers of their children. The hub of *c* is 60, and *c* has two children, so the authority that *c* transmits to *a* is 30; the hub of *d* is 80, and *d* has two children, so the authority that *d* transmits to *a* is 40; and hence, the authority of *a* is 70. The hub of *a* is decided by *f*, *g* and the numbers of their parents. The authority of *f* is 10, and *f* has two parents, so the hub that *f* transmits to *a* is 20; the authority of *g* is 60, and *g* has two parents, so the hub that *g* transmits to *a* is 30; and hence, the hub of *a* is 50. With the same reason, the authority and hub of *b* are 110 and 100, respectively.

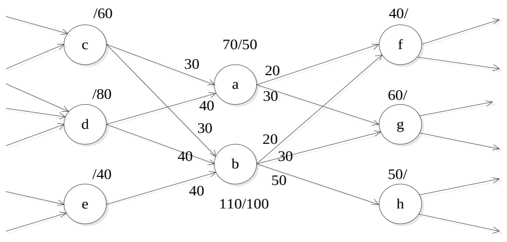

Figure 1. Computation of authority/hub.

B. Random walk model of MCHITS

Given a graph *G*, for any node v_i in *V*, the influence attributes of v_i are denoted with a_i and h_i, and the random walk model of MCHITS is as follows:

- choose a source node *s* randomly, and make the point *current* point to *s*, that is *current*←*s*;
- at each random step, choose a node v_i from the neighbors (including fathers and children) of *current*, that is *current*←v_i;
- repeat the above step *k* times.

The probabilities that random walk passes each node v_i include a_i and h_i. a_i means the probability of stability of v_i that comes from its children, and h_i means the probability of stability of v_i that comes from its fathers. When $k\to\infty$, however we choose *s*, a_i and h_i tend to be constants, and they are the authority and hub of v_i. The set of visited nodes of a random walk is called its fingerprint.

C. Monte Carlo based MCHITS Approximate Algorithms

The Monte Carlo method simulates several (pseudo) random walks, and approximates the probability of stability of each node with visited frequency. At the beginning, we initiate a_i and h_i of each node v_i in *G* with 0.

As each node in a graph contains authority and hub, we denote the probability of node *j* that comes from source node *i* with $X_{i,j}+Y_{i,j}$. $X_{i,j}$ means that the last node of *j* is its father, and $Y_{i,j}$ means that the last node of *j* is its child. The statistics variants $X_{i,j}$ and $Y_{i,j}$ of a graph are in table 1.

TABLE I. MONTE CARLO SAMPLING TABLE

$X_{1,1}$ / $Y_{1,1}$	$X_{1,2}$ / $Y_{1,2}$	…	$X_{1,n}$ / $Y_{1,n}$
$X_{2,1}$ / $Y_{2,1}$	$X_{2,2}$ / $Y_{2,2}$	…	$X_{2,n}$ / $Y_{2,n}$
…	…	⋱	…
$X_{n,1}$ / $Y_{n,1}$	$X_{n,2}$ / $Y_{n,2}$	…	$X_{n,n}$ / $Y_{n,n}$

While sampling, we do a number *R* of random walks from each source node, so $X_{i,j}^{(k)}$ ($1\le k\le R$) are independent with each other, and $Y_{i,j}^{(k)}$ ($1\le k\le R$) are independent with each other, too.

$$\begin{aligned} f_a(X_{s,j}=x) &= P\left(\left[\sum_{t\ge1}^{t\in\text{Odd}} 1_{\{Z_t=j\}}\right]=x \mid Z_0=s\wedge Z_{t-1}=u\wedge(u,j)\in E\right) \quad j=1,\ \cdots,\ |V|; \\ f_h(Y_{s,j}=y) &= P\left(\left[\sum_{t\ge1}^{t\in\text{Even}} 1_{\{Z_t=j\}}\right]=y \mid Z_0=s\wedge Z_{t-1}=u\wedge(j,u)\in E\right) \quad x,\ y=0,\ 1,\ \cdots. \end{aligned} \tag{4}$$

$$\begin{aligned} f_a(X_{i,j}=x) &= P\left(\left[\sum_{t\ge1}^{t\in\text{Odd}} 1_{\{Z_t=j\}}\right]=x \mid Z_0=i\wedge Z_{t-1}=u\wedge(u,j)\in E\right) \quad i,\ j=1,\ \cdots,\ |V|; \\ f_h(Y_{i,j}=y) &= P\left(\left[\sum_{t\ge1}^{t\in\text{Even}} 1_{\{Z_t=j\}}\right]=y \mid Z_0=i\wedge Z_{t-1}=u\wedge(j,u)\in E\right) \quad x,\ y=0,\ 1,\ \cdots. \end{aligned} \tag{5}$$

$$\begin{aligned} f_a(X_{i,j}=x) &= P\left(\left[\sum_{1\le t\le k}^{t\in\text{Odd}} 1_{\{Z_t=j\}}\right]=x \mid Z_0=i\wedge Z_{t-1}=u\wedge(u,j)\in E\right) \quad i,\ j=1,\ \cdots,\ |V|; \\ f_h(Y_{i,j}=y) &= P\left(\left[\sum_{1\le t\le k}^{t\in\text{Even}} 1_{\{Z_t=j\}}\right]=y \mid Z_0=i\wedge Z_{t-1}=u\wedge(j,u)\in E\right) \quad x,\ y=0,\ 1,\ \cdots. \end{aligned} \tag{6}$$

1) Single Source Monte Carlo Approximation of MCHITS

If we apply one single fingerprint to approximate MCHITS, the main idea is that: choose a source node *s* randomly, and make the *current* point to *s*; at each random step, judge the termination condition using random number, and if the termination condition isn't satisfied, continue this random walk; choose a node *next* randomly from *current*'s neighbors, if *next* is the child of *current*, add 1 to a_{next}, else add 1 to h_{next}; repeat the above random step until the algorithm terminates. Repeat this algorithm R times, and thus we can approximate a_i and h_i of v_i by dividing by the total length L of all fingerprints, and the details are in algorithm 1. The probability density function of $X_{s,j}$ and $Y_{s,j}$ is in equation (4), where $1_{\{z_t=y\}}$ equals to 1 when arrive at y at the t step from the source node, and 0 else.

```
Algorithm 1. MC-one.
Input: a graph $G = (V, E)$, stop probability $\varepsilon$;
Output: influence vector $a$, $h$;
1: randomly choose start $s$ from $V$;
2: let $current \leftarrow s$, $totalStep \leftarrow 0$;
3: while ( Random.nextDouble() > $\alpha$ ) do
4:   randomly choose a neighbor $i$ from $current$;
5:   let $next \leftarrow I$, $totalStep \leftarrow totalStep+1$;
6:   if ( $current$, $next$ ) $\in E$ then
7:     let $a_{\text{next}} \leftarrow a_{\text{next}}+1$;
8:   else
9:     let $h_{\text{next}} \leftarrow h_{\text{next}}+1$;
10:   end if
11:   let $current \leftarrow next$;
12: end while
13: $a \leftarrow a/totalStep$, $h \leftarrow h/totalStep$;
```

2) Many Sources Monte Carlo Approximation of MCHITS

In practice, G is always very large, and thus we need to process it with parallel or distributed systems. Under this condition, we can simulate multi random walks at the same time, and thus we can choose multi source nodes. The choosing of source nodes can be decided by the number of processing unit or the number of nodes on graphs. In this paper, we make each node as source node, and do R times random walks. The idea is the same as the single source except the choosing of source nodes. In order to get the same length of all fingerprints, we set the average length of a random walk equal to $L/|V|$, that is $\overline{step} = L/|V|$, and the details are in algorithm 2. The probability density function of $X_{i,j}$ and $Y_{i,j}$ is in equation (5).

```
Algorithm 2. MC-all.
Input: a graph $G = (V, E)$, stop probability $\varepsilon$;
Output: influence vector $a$, $h$;
1: for all $v \in V$ do
2:   let $current \leftarrow v$, $Step_v \leftarrow 0$;
3:   while ( Random.nextDouble() > $\beta$ ) do
4:     randomly choose a neighbor $i$ from $current$;
5:     $next \leftarrow i$, $Step \leftarrow Step+1$;
6:     if ( $current$, $next$ ) $\in E$ then
7:       let $a_{\text{next}} \leftarrow a_{\text{next}}+1$;
8:     else
9:       let $h_{\text{next}} \leftarrow h_{\text{next}}+1$;
10:     end if
11:     $current \leftarrow next$;
12:   end while
13: end for
14: $totalStep \leftarrow \sum_{v\in V} Step_v$ ;
15: let $a \leftarrow a/totalStep$, $h \leftarrow h/totalStep$;
```

3) Many Sources Fixed Length Monte Carlo Approximation of MCHITS

In parallel or distributed systems, the execution time of an algorithm is decided by the longest process. As we take all nodes of a graph as source nodes and the length of fingerprints conforms to the geometric distribution, so a huge amount of fingerprints degrades the efficiency of the algorithm very much. In order to improve the efficiency of the algorithm, we fix the length of each fingerprint with $L/|V|$, and the details are in algorithm 3. The probability density function of $X_{i,j}$ and $Y_{i,j}$ is in equation (6).

```
Algorithm 3. MC-all-k.
Input: a graph $G = (V, E)$, fingerprint length $k$;
Output: influence vector $a$, $h$;
1: for all $v \in V$ do
2:   let $current \leftarrow v$;
3:   for $i$=1 to $k$ do
4:     randomly choose a neighbor $i$ from $current$;
5:     $next \leftarrow i$;
6:     if ( $current$, $next$ ) $\in E$ then
7:       let $a_{\text{next}} \leftarrow a_{\text{next}}+1$;
8:     else
9:       let $h_{\text{next}} \leftarrow h_{\text{next}}+1$;
10:     end if
11:     $current \leftarrow next$;
12:   end for
13: end for
14: $totalStep \leftarrow n \cdot k$;
15: let $a \leftarrow a/totalStep$, $h \leftarrow h/totalStep$;
```

IV. Theoretical Analysis

In random walks of the MCHITS method, we can go to the child or father of the current node, and the transition possibility matrix is $P \bigcup P^{\mathrm{T}}$. According to the PageRank definition, we have $\pi = (P \bigcup P^T) \cdot \pi$, that equals to $\pi = P \cdot \pi + P^T \cdot \pi$. However, in the above random walks, $P \cdot \boldsymbol{h} = P \cdot \pi$ and $P^{\mathrm{T}} \cdot \boldsymbol{a} = P^{\mathrm{T}} \cdot \pi$, so we have $\pi = \boldsymbol{a} + \boldsymbol{h}$.

After all random walks terminate, we can approximate the influence attributes of each node by the following formula:

$$\begin{aligned} \bar{a}_j &= \left[\sum_{k=1}^{R} \sum_{i=1}^{|V|} X_{i,j}^{(k)} \right] \left[\sum_{k=1}^{R} \sum_{i,j=1}^{|V|} (X_{i,j}^{(k)} + Y_{i,j}^{(k)}) \right]^{-1} \\ \bar{h}_j &= \left[\sum_{k=1}^{R} \sum_{i=1}^{|V|} Y_{i,j}^{(k)} \right] \left[\sum_{k=1}^{R} \sum_{i,j=1}^{|V|} (X_{i,j}^{(k)} + Y_{i,j}^{(k)}) \right]^{-1} \end{aligned} \tag{7}$$

The choosing of source nodes influences the efficiency of the Monte Carlo method. We denote the variances of the single source algorithm with $\mathrm{Var}(X_{s,j})$ and $\mathrm{Var}(Y_{s,j})$, and the variances of the many source nodes algorithms with $\mathrm{Var}(X_{i,j})$ and $\mathrm{Var}(Y_{i,j})$, then we have

$$\begin{aligned} \mathrm{Var}(X_{s,j}) &= \frac{1}{|V|} \sum_{i=1}^{|V|} \mathrm{Var}(X_{i,j}) + \frac{1}{|V|} \sum_{i=1}^{|V|} \mathrm{E}^2(X_{i,j}) \\ &- \left[\frac{1}{|V|} \sum_{i=1}^{|V|} \mathrm{E}(X_{i,j}) \right]^2 > \frac{1}{|V|} \sum_{i=1}^{|V|} \mathrm{Var}(X_{i,j}) \end{aligned} \tag{8}$$

With the same reason, we can also have $\mathrm{Var}(Y_{s,j}) > \frac{1}{|V|}\sum_{i=1}^{|V|}\mathrm{Var}(Y_{i,j})$. In brief, the many source Monte Carlo method has a smaller variance, and thus is more efficient than the single source method.

V. Experiments

We validate the efficiency and accuracy of our MCHITS method by the above Monte Carlo based approximate algorithms. The experiment environment is a personal computer with JDK1.6. We employ the JUNG as our toolkit to process graph data. Our graph datasets are Dianping [22] and Gowalla [23]. The Dianping dataset is crawled on the Dianping website, and the Gowalla dataset is a public dataset. Details of the datasets are in Table 2.

TABLE II. Details of datasets.

dataset	#node	#edge	Average degree
Dianping	204,074	926,720	9.08
Gowalla	196,591	1,900,654	19.34

Where #node is the number of nodes, and #edge is the number of edges.

A. Results

As an approximate method of HITS, the purpose of MCHITS is improving the efficiency of HITS, while at the same time, keeping higher accuracy. In the following experiments, we denote the single source Monte Carlo based MCHITS with MC-one, the many sources Monte Carlo based MCHITS with MC-all, and the many sources fixed length Monte Carlo based MCHITS with MC-all-k.

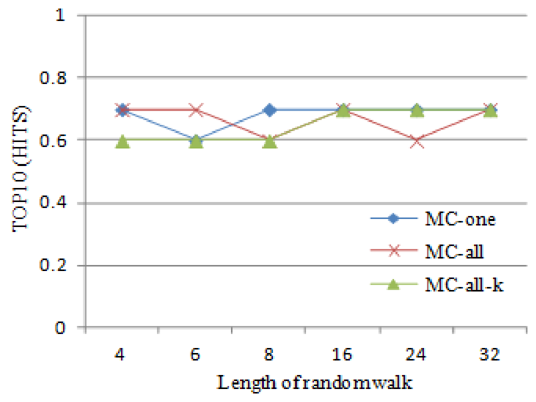

Figure 2. Monte Carlo algorithms comparison (top10, Gowalla).

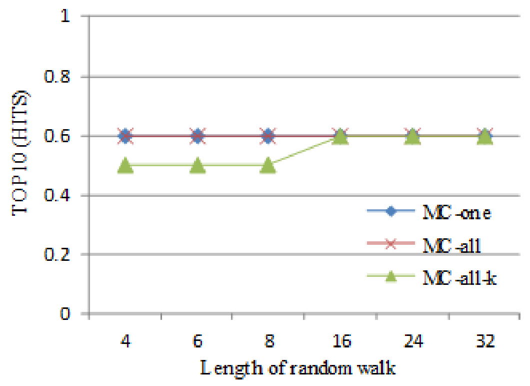

Figure 3. Monte Carlo algorithms comparison (top10, Dianping).

In practice, users only care about the first *m* results. In order to validate the accuracy of our algorithms, we compare the first 10 and 100 results with original HITS results, and observe the proportion of the same results. We observe the accuracy of the three algorithms while changing the length of the walks, and the results are in Figure 2, 3, 4 and 5. Here, the length of random walk is the average visited frequency of each node. From these figures we can see that, the Monte Carlo method of HITS approximation can reach a high accuracy with very short length of random walk.

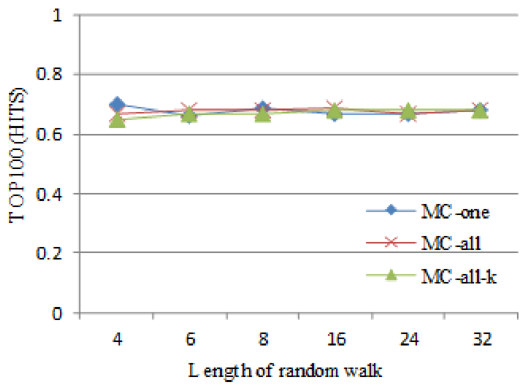

Figure 4. Monte Carlo algorithms comparison (top100, Gowalla).

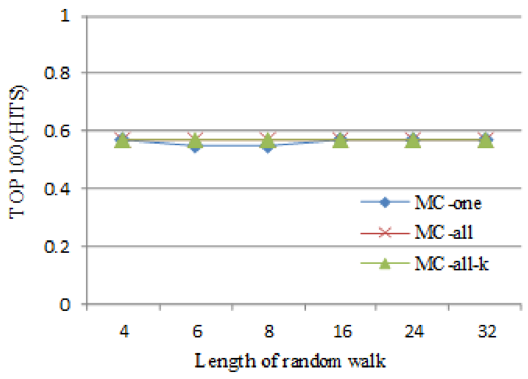

Figure 5. Monte Carlo algorithms comparison (top100, Dianping).

In order to compare the accuracy of these algorithms, we set the length of fingerprints at 6. We compare our Monte Carlo based HITS computation with SALSA and ACHITS, and the results are in figure 6 and 7. For the ACHITS algorithm works better on graphs with bigger average degree [2], it has better accuracy in the Gowalla dataset than the Dianping dataset for both the top 10 and the top 100 results. The SALSA algorithm has the same efficiency in both datasets, but it has low accuracy. Our proposed Monte Carlo method works best in both datasets for both the top 10 and the top 100 results.

The Monte Carlo based algorithms can reach stabilization in a small visited frequency of node. We set 6 as the visited frequency, and compare our algorithms with SALSA and ACHITS, and the result is in Fig. 2. As can be seen from the figure that the accuracy is different in the ACHITS algorithm, and it suits graphs of big average degree; our Monte Carlo based algorithms is more accurate in both kind of graphs.

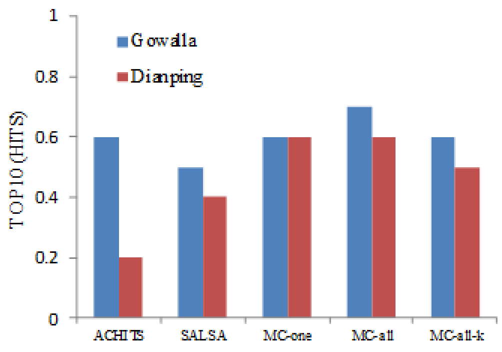

Figure 6. Similarity comparison (top10).

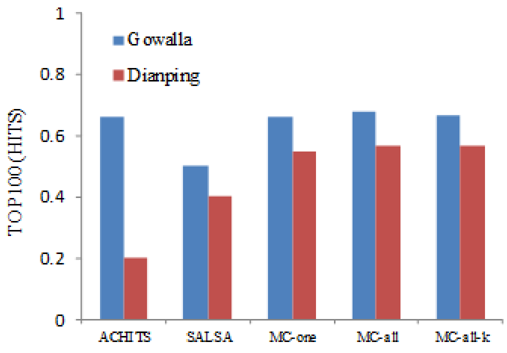

Figure 7. Similarity comparison (top100).

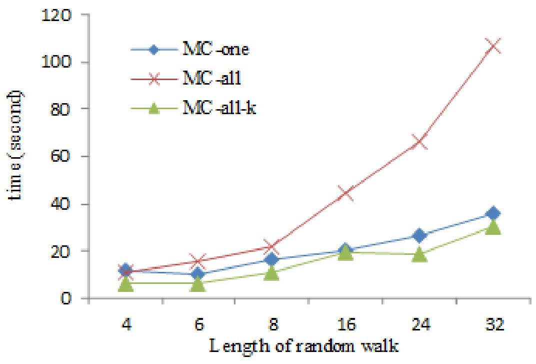

Figure 8. Monte Carlo algorithms time comparison (Gowalla).

The efficiency of the Monte Carlo based algorithms relates to the specified random walks. We observe their efficiencies with different visited frequencies. Fig. 3 describes the detail of them. The randomness affects the efficiency of the designed algorithm. In MC-all, we have many fingerprints, and each of them is random, so the execution time is longest. The growth of MC-one and MC-all-k is almost linear. The MC-all-k algorithm has many fingerprints, the length is constant. Though the MC-one algorithm has only one fingerprint, the length is random, and thus it has longer execution time. We set 6 as the average length of fingerprint, and compare the execution time of HITS, SALSA, ACHITS and the proposed Monte Carlo based algorithms. As can be seen from Fig. 4 that the proposed Monte Carlo based algorithms have similar or even smaller execution time than ACHITS, which is much faster than HITS and SALSA.

In a word, the Monte Carlo based algorithms for MCHITS can not only approximate the HITS ranking on graphs much faster, but also has higher accuracy. While designing the Monte Carlo based algorithms, randomness is a key factor. Higher randomness will bring higher accuracy, but longer execution time.

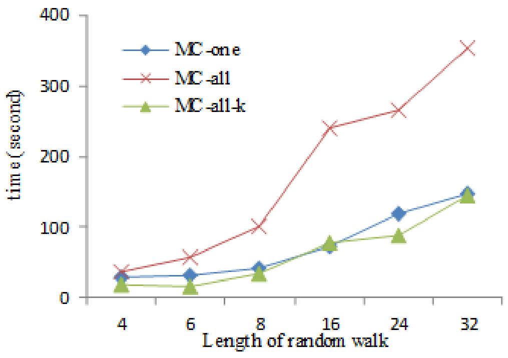

Figure 9. Monte Carlo algorithms time comparison (Dianping).

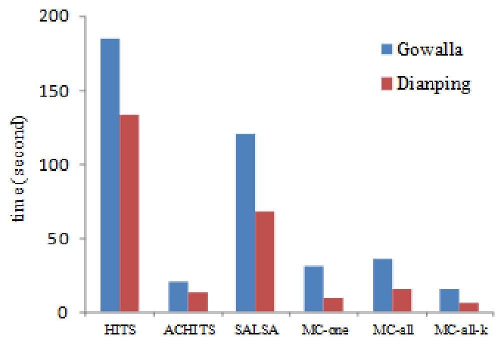

Figure 10. Algorithms time comparison.

VI. Conclusion and Future Work

With the rapid growth of data on World Wide Web, how to return users with related contents is one of the most important researches in information retrieval. This paper defines MCHITS, a random walk model for HITS, and proposes three Monte Carlo based approximate algorithms for MCHITS. Experiments show that the Monte Carlo based approximate computing of the HITS ranking reduces computing resources a lot while keeping higher accuracy, and is significantly better than related work.

The experiment is carried on small datasets, and more challenge must be solved while facing massive data. As the same time, more and more Webpages join and depart, and this makes the World Wide Web a streaming graph. How to design incremental algorithm, how to design efficient parallel algorithm and how to make up the deficiencies of the HITs algorithm will be our future work.

Acknowledgement

This work was supported in part by the National Significant Science and Technology Special Project of China (Nos. 2011ZX03002-004-01) and the National

Natural Science Foundation of China (No.91118008, No.61202117).

REFERENCES

[1] Page L, Brin S, Motwani R, et al. The PageRank Citation Ranking: Bringing Order to the Web. *Technical Stanford InfoLab*, 1999.

[2] Kleinberg J M. Authoritative sources in a hyperlinked environment. *Journal of the ACM (JACM)*. 1999, 46(5) pp. 604-632.

[3] Jeh G, Widom J. SimRank: a measure of structural-context similarity. *In IEEE* pp. 2002. 538-543.

[4] Sarl O S T, Bencz U R A A, Csalog A Ny K, et al. To randomize or not to randomize: Space optimal summaries for hyperlink analysis. *In IEEE* pp. 2006. 297-306.

[5] Lempel R, Moran S. The stochastic approach for link-structure analysis (SALSA) and the TKC effect. *Computer Networks*. 2000, 33(1) pp. 387-401.

[6] Mirzal A, Furukawa M. A Method for Accelerating the HITS Algorithm. *CoRR*. 2009, abs/0909.0572.

[7] Avrachenkov K, Litvak N, Nemirovsky D, et al. Monte Carlo methods in PageRank computation: *When one iteration is sufficient*. 2005.

[8] Fogaras D, R A Cz B, Csalog A Ny K, et al. Towards scaling fully personalized pagerank: Algorithms, lower bounds, and experiments. *Internet Mathematics*. 2005, 2(3) pp. 333-358.

[9] Ng A Y, Zheng A X, Jordan M I. Stable algorithms for link analysis. *In IEEE*: 2001 pp. 258-266.

[10] Haveliwala T H. Topic-sensitive pagerank: A context-sensitive ranking algorithm for web search. Knowledge and Data Engineering, *IEEE Transactions on*. 2003, 15(4) pp. 784-796.

[11] Balmin A, Hristidis V, Papakonstantinou Y. Objectrank: Authority-based keyword search in databases. *In IEEE*: 2004 pp. 564-575.

[12] Bahmani B, Chowdhury A, Goel A. Fast incremental and personalized PageRank. *Proceedings of the VLDB Endowment*. 2010, 4(3) pp. 173-184.

[13] Bahmani B, Chakrabarti K, Xin D. Fast personalized pagerank on mapreduce. *In IEEE*: 2011 pp. 973-984.

[14] Dean J, Ghemawat S. MapReduce: Simplified data processing on large clusters. *Communications of the ACM*. 2008, 51(1) pp. 107-113.

[15] Gollapudi S, Najork M, Panigrahy R. Using bloom filters to speed up hits-like ranking algorithms. *Algorithms and Models for the Web-Graph.* 2007 pp. 195-201.

[16] Deng H, Lyu M R, King I. A generalized Co-HITS algorithm and its application to bipartite graphs. *In IEEE*: 2009 pp. 239-248.

[17] Li L, Shang Y, Zhang W. Improvement of HITS-based algorithms on web documents. *In IEEE*: 2002 pp. 527-535.

[18] Bharat K, Henzinger M R. Improved algorithms for topic distillation in a hyperlinked environment. *In IEEE:* 1998 pp. 104-111.

[19] Nomura S, Oyama S, Hayamizu T, et al. Analysis and improvement of HITS algorithm for detecting Web communities. *In IEEE*: 2002 pp. 132-140.

[20] Romero D, Galuba W, Asur S, et al. Influence and passivity in social media. *Machine Learning and Knowledge Discovery in Databases*. 2011 pp. 18-33.

[21] Asano Y, Tezuka Y, Nishizeki T. Improvements of HITS algorithms for spam links. *In IEEE*: 2007 pp. 479-490.

[22] Jin Z, Shi D, Wu Q, et al. LBSNRank: personalized pagerank on location-based social networks. *Proceedings of the 2012 ACM Conference on Ubiquitous Computing. ACM*, 2012 pp. 980-987.

[23] Cho E, Myers S A, Leskovec J. Friendship and mobility: User movement in location-based social networks. *In IEEE*: 2011 pp. 1082-1090.

Zhaoyan Jin, born in 1981, received his B.S. degree of communication and M.S. degree of Computer Science and Technology from the National University of Defense Technology of China in 2004 and 2008, respectively.

He is currently a Ph.D. candidate by research in Computer Science and Technology in the National University of Defense Technology of China. His research interests include Cloud computing, social network, location-based service and data mining.

Dianxi Shi, born in 1966, Ph.D. Associate Professor. His research interests focus on distributed computing, middleware and mobile cloud computing.

Quanyuan Wu, born in 1941, Professor, PH.D. Supervisor. His research interests include distributed computing, network security and artificial intelligence.

Hua Fan, born in 1984, is currently a Ph.D. candidate by research in Computer Science and Technology in the National University of Defense Technology of China. His research interests include Cloud computing, social network, Internet of Things.