International Journal of Security and Its Applications
Vol.9, No.11 (2015), pp.231-248
http://dx.doi.org/10.14257/ijsia.2015.9.11.22

Improved Subgraph Estimation PageRank Algorithm for Web Page Rank

Lanying Li[1], Qiuli Zhou[1], Yin Kong[1] and Yiming Dong[1]
[1]The college of Computer Science and Technology, Harbin University of Science and Technology, Harbin 150080, China
lulu08521@sina.com

Abstract

The traditional PageRank algorithm can't efficiently dispose large data Webpage scheduling problem. This paper proposes an accelerated algorithm named topK-Rank .It is based on PageRank on the MapReduce platform. Owing to this algorithm ,Top k nodes can be found efficiently for a given graph without sacrificing accuracy. It can iteratively estimate lower/upper bounds of PageRank scores, and construct subgraphs in each iteration by pruning unnecessary nodes and edges. Theoretical analysis shows that this method guarantees result exactness. Experiments show that it can find top k nodes much faster than the existing approaches.

Keywords: *web data; webpage scheduling; PageRank algorithm; MapReduce*

1. Introduction

PageRank[1] is a commonly used algorithm for search engines[2], which is based on links between pages to calculate web page rank. But traditional PageRank has high computational complexity, which has led to the introduction of acceleration approaches. Generally speaking, there are three main acceleration algorithms: Linear Algebraic, Dynamic Programming, and Monte-Carlo.

Reference [3] proposed a linear algebraic scheme, which stops the iteration once the PageRank score of the node converges. However, this approach may be not convergence, and results may be not correct. Reference [4] studied another linear algebraic scheme, they applied Krylov subspace methods, which is faster than the original approach. However, the convergence is not stationary, so PageRank results behave erratically. As to the problems that the calculation of incremental change for the web graph affected only the local PageRank, reference [5] presented an approximation mechanism, but this kind of method is difficult to improve the quality of practical application. Reference [6] put forward the Monte-Carlo approach. It can approximately compute the top k nodes named top-k in ad-hoc style since they perform random walks on a given graph. However, this approaches require the number of random walks to be set, and need to find a balance point between efficiency and the quality of approximation.

With the rapid development of computer technology, the PageRank algorithm parallelization has become inevitable. In Hadoop cloud computing environment, reference [7] conducted parallel computing on the PageRank algorithm. Compared with the single serial, the algorithm performance was improved, and the time complexity was reduced. Reference [8] proposed a parallel approach based on the block structure division, which decreased the number of map and reduce operations, reduced the I/O transfers overhead, and improved the efficiency of computing. To obtain a more precise quantitative results, reference[9] introduced a state transition matrix to realize iteration of the user importance ranking. The result is not only a reasonable reflection of the number of user's fans, but also take the quality of user's fans into account and improved the search ranking. In a word, existing parallel algorithms usually require a lot of iterative calculation, frequently access to HDFS and too much cluster communication, but this is time-consuming.

To overcome the limitations of existing methods, we select MapReduce[10] as a framework for parallel computing, and use the open source Apache Hadoop [11] platform to achieve a novel algorithm: topK-Rank. It can rapidly and accurately find top-k node set. To reduce computation cost,(1)we estimate lower/upper bounds of PageRank scores in each iteration by subgraphs,(2)in order to identify top-k nodes efficiently and reduce the number of iterations, we dynamically construct subgraphs. TopK-Rank has the following characteristics:

(1) Fast: Based on the above idea, it is faster than the previous approaches
(2) Exact: Without sacrifice accuracy, it returns the exact top-k nodes
(3) Flexible: Without any pre-computation, it can effectively handle peer to peer search for any arbitrary graphs.
(4) Parameter-free: It does not require any inner-parameters, which provides a user with a simple solution to PageRank-based applications.

2. PageRank

2.1. The Description of PageRank

The PageRank algorithm is described as follows:

$$p_i = dWp_{i-1} + (1-d)e \qquad (1)$$

Among them, n: the total number of pages, $e = 1/n$, p_i: different pages $p_1, p_2, p_3, \ldots, p_n$, W: column normalized adjacency matrix, d: damping coefficient, which is between 0 and 1. Google often sets to 0.85,$1-d = 0.15$: the probability of users to stop clicking and random jumping to a new URL.

2.2. Parallel Implementation of PageRank

Principle of PageRank algorithm, Briefly, is to realize parallelization by matrix calculation. Firstly, store the column of the adjacency matrix according to the data line, and then repeat iterations: calculate the matrix eigenvalues, until it is convergence or get the expected number of iterations. Parallelization process is divided into map and reduce phase.

The Map phase is completed the mapping of key value pairs. The input parameters is key value pairs which is component of <Adjacency matrix , The value of PR>.Among the column of the adjacency matrix is the source page, the row of it is the target page, and the initial value of PR is 1-d;After the internal treatment ,the output parameters is key value pairs which is component of <Row number of PR , The multiplicative summation formula of the adjacency matrix and the PR value>.

The Reduce phase does the merge operation by key which has the same key value. It regards the output of the map process as input ,after processing, the output parameters is key value pairs which is component of <Row number of PR , The final PR value>.

After the completion of the Reduce phase, we read the file name and the PR value of the last iteration results, ranking the PR value from big to small, and finally get the web ranking corresponds to the PR value.

3. TopK-Rank

3.1. A Full Description of TopK-Rank Algorithm

The core idea of topK-Rank is to reduce computation cost. Instead of using the whole graph to compute PageRank scores, this method makes cutting rules by pruning unnecessary nodes and edges, and obtains subgraphs. By computing the estimations of candidate nodes in the subgraphs, the required PR value can be obtained finally. Table 1 shows the main symbols and their definitions in our paper.

Table 1. Definition of Main Symbols

symbol	definition	symbol	definition
N	Number of nodes in the graph	M	Number of edges in the graph
T	Number of iterations by the existingl approach	k	Number of final answer nodes
ε_i	k-th highest lower estimation in the i-th iteration	G	Given graph
V	nodes in G	E	edges in G
C	candidate nodes set	R	reachable nodes set
W	N*N column normalized adjacent matrix of G	p	N*1 PageRank vector
$\overline{p_i}$	N*1 upper bounding PageRank vector	$\underline{p_i}$	N*1 lower bounding PageRank vector

3.1.1. Lower and Upper Estimations: To obtain subgraphs, the estimations can be computed for the candidate node set C_i in the i -th $(i=0,1,2,...,n)$ iteration. To compute the upper estimation, topK-Rank uses 3 parameter as follows.

(1) R_i , the set of reachable nodes to any node in C_i . If there is a path from node u to v, we can say node u is reachable to v.

(2) $\overline{W}$, a $N*1$ vector of the maximum edge weights, and each element is given by $\overline{W}[u]=\max\{W[u,v]:v\in V\}$.

(3) r_i , the $N*1$ probability vector, and the length of its random walk is i .Among, $r_i=W^ie$, where W is the adjacent matrix. If $i=0$, then $r_i=W^ie$, where i is the identity matrix.

Respectively, We define the lower and upper bounding PageRank vectors in the i -th iteration as $\overline{p_i}$ and $\underline{p_i}$.

Definition 1(Lower estimation) In the $i-th$ iteration, the lower estimation defines as follows:

$$\underline{p_i}=(1-d)\sum_{j=0}^{i}d^jr_j \qquad (2)$$

Definition 2(Upper estimation) In the $i-th$ iteration, the upper estimation defines as follows:

$$\overline{p_i}=\left(1-d\right)\sum_{j=0}^{i}d^jr_j+d^{i+1}r_j+\Delta_i\sigma_i\overline{w} \qquad (3)$$

Among, $\sigma_i=d^{i+1}(1-d)^{-1}$ and Δ_i is computed as follows:

$$\Delta_i=\begin{cases}1 & (i=0)\\ \sum_{u\in R_i}\Delta_i[u] & (i\neq 0)\end{cases} \qquad (4)$$

where $\Delta_i\left[u\right]=\max\left\{r_i\left[u\right]-r_{i-1}\left[u\right],0\right\}$

According to Definition 1 and 2, We can recursively compute the lower and upper estimations in each iteration by using the random walk probabilities.

3.1.2. Subgraph: To obtain the top k nodes, C_i can be computed recursively. If the number of candidate nodes is k, the iterations is stopped. We compute the estimations only for candidate nodes in subgraphs, which are dynamically obtained in each iteration.

Definition 3 (candidate nodes)

$$C_i = \begin{cases} V & (i=0) \\ \left\{u \in V : \overline{p_{i-1}}[u] \geq \varepsilon_{i-1}\right\} & (i \neq 0) \end{cases} \tag{5}$$

Where ε_{i-1} is the k-th highest lower estimation in the previous $(i-1)-th$ iteration. The above equation shows: If the upper estimation of the node is not lower than ε_{i-1}, then it is a candidate node. Otherwise, the exact PR score of the node must be lower than ε_{i-1}, which is irresponsible.

Definition 4(update of candidate nodes) If $i \neq 0$, then C_i can be computed incrementally in each iteration by equation(6).

$$C_i = \{u \in C_{i-1} : \overline{p_{i-1}}[u] \geq \varepsilon_{i-1}\} \tag{6}$$

Definition 5 (subgraphs) If $i=0$, then V_0 and E_0 are defined as V and E respectively· If $i \neq 0$, then $V_i = R_i$, $E_i = \{(u, v) \subseteq E : u \subseteq R_i, v \subseteq R_i\}$, where (u, v) represents the edge from node u to v.

Definition 6(Incremental estimations) the upper and lower estimations are incrementally computed as follows:

$$\underline{p_i}[u] = \begin{cases} (1-s)/N & (i=0) \\ \underline{p_{i-1}}[u] + (1-s)s^i r_i[u] & (i \neq 0) \end{cases} \tag{7}$$

$$\overline{p_i}[u] = \begin{cases} 1/N + s(1-s)^{-1}\bar{W}[u] & (i=0) \\ \underline{p_{i-1}}[u] + s^i r_i[u] + \Delta_i \sigma_i \bar{W}[u] & (i \neq 0) \end{cases} \tag{8}$$

If $i \neq 0$, then $r_i[u] = \sum_{v \in V_i} W[u,v] r_{i-1}[u]$, where $r_0 = e$.

This definition indicates that if $i=0$, then the estimation of a node are obtained by the probability d, the number of nodes and the edge weights, otherwise, we can incrementally update the lower/upper estimations from the lower estimation of the previous iteration.

3.2. The MapReduce Implementation of TopK-Rank

(1) Map process: Map process looks for R_i for the specified key according to the depth-first algorithm. We calculate $r_i[u]$ for each target node u in R_i, and then output key/value pair < target node, $r_i[u]$ >.

(2) Reduce process: Firstly, we respectively calculate the upper and lower estimations of PageRank according to the formula 7 and 8. Then judge the new candidate set by the formula 6 for each page u .Finally, the results are saved in the HDFS and used for the next iteration.

(3) We combine the intermediate results with the link subgraph results generated by before, then iteratively implement the improved parallel PageRank algorithm, and statistics the length of candidate node set, finally we get the top-k nodes.

In order to obtain the final desired top-k set of nodes, the time complexity of the topK-Rank algorithm is $O((n+m+\log c \log k)t)$.

3.3. The Time Complexity of the Algorithm

In order to obtain the final desired top-k set of nodes, the time complexity of the topK-Rank algorithm is $O((n+m+\log c\log k)t)$.

Proof: topK-Rank first constructs the subgraphs by depth-first search at the cost of $O((n+m)t)$. It computes the random walk probabilities for each node in the subgraphs, which needs $O((n+m)t)$; Since it needs $O(1)$ time to compute the lower/upper estimations of a node in each iteration, the estimations of the candidate nodes are obtained at the cost of $O(ct)$. It computes ε_i from the candidate nodes by the lower estimations, which requires $O(\log c\log k)$.This is because(1)it updates the lower estimation in each iteration from the candidate nodes, which needs $O(\log k)$;(2)the expected number of update is $O(\log c)$ by randomly accessing the candidate nodes; By using ε_i and the lower estimations, C_{i+1} is obtained at $O(ct)$ from C_i. Therefore, the time complexity of the topK-Rank algorithm is $O((n+m+\log c\log k)t)$.

4. Experiment and Result Analysis

4.1. Experimental Platform and Data

Our experiment uses six PCs. The configuration lists as follows: CPU is Intel Xeon X3330, memory capacity is 4GB, the hard disk capacity is 1TB. Then through 10M switch, the Hadoop cloud environment is built. One of six PCs is the Master service node of NameNode and JobTracker, the others is the Slave service node of DateNode and TaskTracker. Each node is installed with Ubuntu14.04, Hadoop1.2.1, JDK1.8, and the development environment adopts eclipse3.6, hadoop1.2.1 plug-in and maven3.2.3. The experiment data set use graph data which provided by USA social network study platform, as shown in Table 2.

Table 2. Experimental Data Set

dataset	number of nodes	number of edges	description
P2P	$6.26*10^4$	$1.48*10^5$	a snapshot of the Gnutella peer-to-peer file sharing network.
Web	$3.26*10^5$	$3.22*10^6$	the result of the Italian CNR domain crawl
Wikipedia	$2.39*10^6$	$5.02*10^6$	Wikipedia page

4.2. Experiment and Result Analysis

4.2.1. Comparison of Iteration Numbers: Instead of using the whole graph to iteratively compute the PageRank score of each node, our methods adopts subgraphs to compute the estimations until the numbers of candidate nodes satisfy the final demand. Firstly, subgraphs are smaller than the given graphs, which can greatly reduce the iteration numbers. Secondly, the candidate nodes obtained by subgraphs is monotonic decreasing, which can rapidly decrease the numbers of nodes and edges. Table 3 details the inner-parameters of each data set where k = 50. Note that they are automatically set by the given graphs and topK-Rank.

Table 3. Result of Iteration Numbers where k=50

Data set	P2P	Web	Wikipedia
N	$6.26*10^4$	$3.26*10^5$	$2.39*10^6$

n	$4.69*10^4$	$2.70*10^5$	$6.29*10^5$
c	$3.16*10^4$	$1.49*10^5$	$4.00*10^5$
M	$1.48*10^5$	$3.22*10^6$	$5.02*10^6$
m	$1.20*10^5$	$3.06*10^6$	$2.44*10^6$
T	18	116	97
t	7	26	15

4.2.2. Comparison of Efficiency: In Figure 1, the results of topK-Rank are indicated by "topK-Rank(k)" where k is the number of the final nodes. The previous study shows that the iterations are terminated when the residual 1-norm dropped below 10^{-10} in the original approach, and the PR scores of all nodes must be computed. Therefore, the number of the final nodes does not affect the time complexity

.

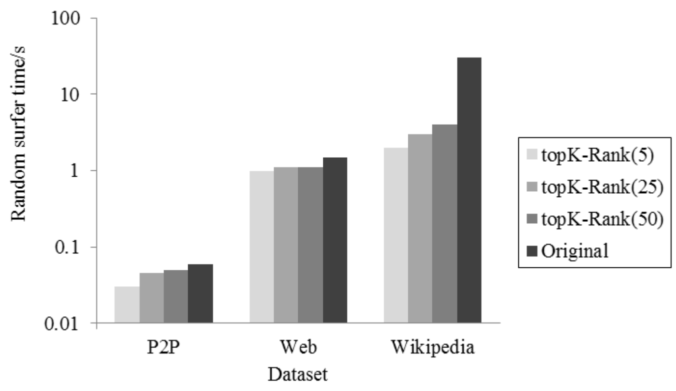

Figure 1. Comparison of the Search Yime

Figure 1 shows topK-Rank is faster than the traditional method, which respectively cut the search time from the traditional method by up to 42%, 72%, 92% for each data set. When the size of graphs increases, topK-Rank can efficiently find the top-k nodes. The existing method iteratively computes PageRank scores for the whole graphs until the PR scores convergence, and its time complexity is $O((N+M)T)$. Respectively, topK-Rank computes the estimations by subgraphs until the number of candidate node is k, and its time complexity is $O((n+m+\log c \log k)t)$.

4.2.3. Comparison of Exactness: The biggest advantage of topK-Rank algorithm is that the final output of the result is the same as the traditional method. In order to prove it, we compared "stop MC complete path stopping in hanging nodes" which was proposed by Avrachenkov *et al*[12].It uses the random walks to sum the total number of visits to the node , and approximates the PageRank score of each node. The number of random walks will affect the search time and approximation accuracy. Therefore, we use a variety of random steps to perform comparative experiments. The precision and search time of data set P2P is given in Figure 2 and Figure3 respectively, where k=50. Figure 2 regards the accuracy as accurate measurement, and Figure 3 regards the wall clock time to evaluate the efficiency of each method.

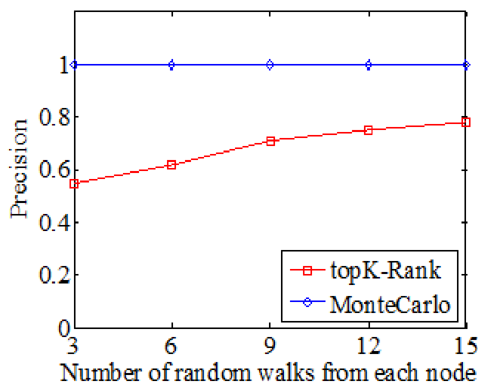

Figure 2. Precision vs. Random Walks

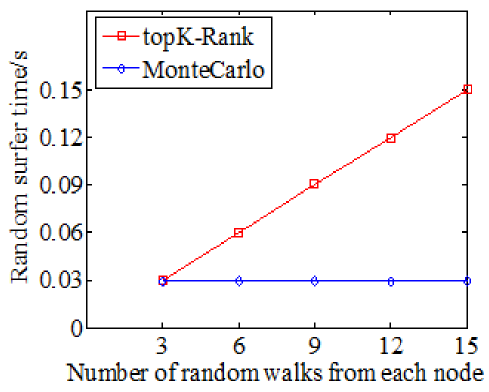

Figure 3. Speed vs. Random Walks

Figure 2, shows that on the one hand, the result of topK-Rank is same as the previous approaches, so the accuracy is one, on the other hand, with the increase of the random steps, although Monte Carlo method can improve accuracy, but the accuracy will reach a peak. The final result shows that Monte Carlo method can't accurately find the top-k nodes, while topK-Rank can output the accurate results. Figure 3 indicates that the search time of Monte Carlo method increases with the increase of the random steps, while topK-Rank is independent of it. In a word, topK-Rank is superior to Monte Carlo in both speed and accuracy.

In summary, when dealing with large-scale graph data, web page rank algorithm based on improved PageRank is more realistic applications, which is high efficient and high accurate.

5. Conclusion

This paper proposed a web page rank based on improved PageRank, named topK-Rank, which can ensure the accuracy of the results. It prunes unnecessary nodes and edges according to the lower and upper estimations and dynamically constructs subgraphs to find the final top-k nodes in each iteration. Experiment results shows that our algorithm is better than the existing approaches, which can be more efficiently when processing many PageRank-based applications. However, there are some challenges, such as the scale of web graphs may exceed the main memory capacity, and block algorithm is also an interesting and challenging problem.

Acknowledgements

Supported by Scientific Research Fund of Heilongjiang Provincial Education Department（NO：12531107).

References

[1] F. Zhenming. "The core of Google core -- PageRank algorithm[J]. Computer Technology and Development", vol. 16, no. 7, pp. 82--84(2006)

[2] D. Huafu, Z. Hongbo. "A Multi-agent Based Personalized Meta- search Engine Using Automatic Fuzzy Concept Networks"[J]. Journal of Harbin University of Science and Technology, vol. 19, no. 02, **(2014)**, pp. 31--35

[3] S. Kamvar, T. "Haveliwala. Adaptive Methods for the Computation of PageRank: Linear Algebra and its Applications[J]", Special Issue on the Conference on the Numerical Solution of Markov Chains, vol. 13, no. 6, **(2008)** , pp. 11—15..

[4] D. Gleich, P. Zhukov. "Fast Parallel PageRank: A Linear System Approach. Research of mining the mutative knowledge with extension data mining[J]", Engineering Sciences,vol.11, N\no. 8, **(2007)** m pp. 70-78

[5] F . McSherry, "A Uniform Approach to Accelerated PageRank Computation[J]. Princeton University Press, vol. 35, no. 8, **(2012),** pp. 17-20.

[6] J. Kai, G. Jihong. “Keywords search based on random walk with restart model on graphs”, Computer Engineering, vol. 37, no. 3, **(2011)**, pp. 68—71.

[7] P.Yu, X. Yang, Z. Bo. “The realization of parallel PageRank algorithm based on MapReduce[J]”. Computer Engineering, vol. 40, no. 2, **(2014)**, pp 31—35.

[8] Z. Yong, Y. Chuanye, W. Chongzheng. “Study on the optimization of PageRank algorithm based on MapReduce[J]”., Computer Engineering, vol. 31, no. 2, **(2014)**, pp. 431—434.

[9] L. Qiushi, W. Yilei, F. Lei. “Study on the optimization of PageRank algorithm based on MapReduce[J]”, Computer Engineering, vol. 32, no. 11, **(2012)**, pp. 2989—299.3

[10] Q. Xiongpai, W. Huiju, D. Xiaoyong, *et al*. “Big Data Analytics ---Competition and Symbiosis between RDBMS and MapReduce[J]”., Journal of software, vol. 23, no. 01, **(2012)**, pp. 32—45.

[11] W. Yufeng, L. Yi. “Study on data access control mechanism based on Hadoop platform[J]”, Engineering and application of computer, vol. 50, no. 22, **(2014)**, pp. 54—258.

[12] K. Avrachenkov, N. Litvak, D. Nemirovsky, “Monte Carlo Method sin PageRank Computation”, When One Iteration is Sufficient[C]. SIAM J Press, **(2007)** , pp. 42—51.