Monte Carlo Based Incremental PageRank on Evolving Graphs

Qun Liao[1], ShuangShuang Jiang[2], Min Yu[1], Yulu Yang[1(✉)], and Tao Li[1]

[1] College of Computer and Control Engineering, Nankai University, Tianjin, China
liaoqun@mail.nankai.edu.cn, yumin2021@163.com, {yangyl,litao}@nankai.edu.cn
[2] Alibaba Cloud (Aliyun), Beijing, China
highfly@mail.nankai.edu.cn

Abstract. Computing PageRank for enormous and frequently evolving real-world network consumes sizable resource and comes with large computational overhead. To address this problem, IMCPR, an incremental PageRank algorithm based on Monte Carlo method is proposed in this paper. IMCPR computes PageRank scores via updating previous results accumulatively according to the changed part of network, instead of recomputing from scratch. IMCPR effectively improves the performance and brings no additional storage overhead. Theoretical analysis shows that the time complexity of IMCPR to update PageRank scores for a network with m changed nodes and n changed edges is O((m+n/c)/c), where c is reset probability. It takes O(1) works to update PageRank scores as inserting/removing a node or edge. The time complexity of IMCPR is better than other existing state-of-art algorithms for most real-world graphs. We evaluate IMCPR with real-world networks from different backgrounds upon Hama, a distributed platform. Experiments demonstrate that IMCPR obtains PageRank scores with equal (or even higher) accuracy as the baseline Monte Carlo based PageRank algorithm and reduces the amount of computation significantly compared to other existing incremental algorithm.

Keywords: PageRank · Web mining · Incremental computing · Monte Carlo algorithm · Parallel and distributed processing

1 Introduction

PageRank plays an important role in Web search, social network analysis and many other application fields [1]. Nowadays, the volume of data in Internet and social networks are tremendous and evolving frequently. Computing PageRank for a large and evolving graph cost huge computational resources. Recomputing from scratch is impractical due to its considerable overhead. Many incremental PageRank algorithms are designed to improve the performance of PageRank computation for dynamic graphs. Algorithm proposed in [2] is one of the most efficient state-of-art algorithms. However, its storage overhead, due to storing all the random walk segments in previous computation, is a limitation which cannot be overlooked.

J. Kim et al. (Eds.): PAKDD 2017, Part I, LNAI 10234, pp. 356–367, 2017.
DOI: 10.1007/978-3-319-57454-7_28

Aiming to compute PageRank for evolving graph efficiently, we propose IMCPR, a novel incremental PageRank algorithm based on Monte Carlo method in this paper. Inspired by previous works [2, 3], our proposed algorithm reuses pervious PageRank scores and updates them incrementally. The proposed avoids a large amount of recomputation and improves the performance significantly.

The most important characteristic of our algorithm is that it stores no previous random walk segments at all, which brings no extra storage overhead. Theoretical analysis also proves that the time complexity of our newly proposed algorithm is lower than other existing related algorithms for most real-world graphs. Moreover, it is also proved that IMCPR performs as good as the original Monte Carlo method in PageRank computation [4] in accuracy. Evaluations based on experiments of real-world graphs demonstrate that IMCPR improves the performance of PageRank significantly compared to other existing PageRank algorithms.

2 PageRank and Related Work

2.1 PageRank

Let $G = (V, E)$ be an unweighted directed graph, where V is the set of nodes and E is the set of edges. $|V|$ is the number of nodes and $|E|$ is the number of edges. For an arbitrary node j, $N(j)$ donates the set of j's outgoing neighbors. $|N(j)|$ is the number of node j's outgoing neighbors. Let A be the transition matrix, where $A(i,j) = 1/|N(i)|$ if and only if there is a direct edge $e(i, j) \in E$, and $A(i,j) = 0$ otherwise. Let π be a vector consisted of PageRank scores of all nodes in V. π is defined as Eq. (1), where α is teleport probability and θ is a vector consisted of fraction $1/|V|$.

$$\pi = \alpha A\pi + (1 - \alpha)\theta \tag{1}$$

The definition of PageRank also has an interpretation based on random walk simulation. Consider a random walk simulation on graph G defined as follows: do R random walks starting from each node of G, for a random walker, at each step it stops with probability c (here we call c as the reset probability and $c = 1 - \alpha$), or jumps to a random chosen outgoing neighbor of current node with probability α. Assume for each node v, $X(v)$ is the total number of times that all random walk segments who visits it. The approximate PageRank of node v $\tilde{\pi}(v)$ is defined as Eq. (2).

$$\tilde{\pi}(v) = cX(v)/(R|V|) \tag{2}$$

Power Iteration [1] is a fundamental algorithm for PageRank computation which computes qualified solution of Eq. (1) iteratively. It is easy to understand and implement, but it's not efficient enough in dealing with massive graph. Many improved algorithms based on this iteration method are well discussed in [5]. Another group of fundamental algorithms are Monte Carlo based algorithms [2, 4]. They simulate the random walks defined above and get accurate approximations efficiently. It is proved that even when $R = 1$, the approximations of important nodes are accurate enough for many applications [4]. These algorithms are also easy to be parallelized.

2.2 Related Work

Bahmani et al. have provided a detailed list of incremental PageRank algorithms in [2]. However, for completeness and for comparison with our own results, we concentrate on reviewing the latest PageRank algorithms for evolving graphs. There are two main categories of incremental PageRank algorithms.

- Aggregation Algorithms

These algorithms [3, 6–8] are based on the idea of graph partition and aggregation. They partition the graph into several sub-graphs and try to limit the affect of changed part of graph in the level of sub-graphs. These methods help to reduce unnecessary recomputation. However, the limitations of these algorithms are high computational load for aggregation, difficulty in partitioning real-world graphs efficiently and unstable performance depending on partitioning. A lot of evidences were discussed in detail in [2].

- Monte Carlo Based Algorithms

The most efficient incremental Monte Carlo based PageRank algorithm was proposed in [2], whose time complexity is $O(|V|\ln(n)/c^2)$ to update PageRank scores as n edges arrivals in a graph [2, 9]. Though it is efficient to update PageRank, the large storage cost for all the random walk segments in history limits the application of this algorithm for large graphs.

Our proposed algorithm doesn't suffer from the shortages of aggregation based algorithms. It handles evolving graphs with nodes and edges inserted and/or removed efficiently. Comparing to the state-of-art algorithm in [2], it requires no extra storage overhead and performs a lower time complexity to update PageRank for most of real-world graphs.

3 Incremental Monte Carlo Method for Pagerank (IMCPR)

We compute approximate PageRank scores according to Eq. (2) as initial solution. As graph evolves, IMCPR updates PageRank based on reusing previous PageRank scores and starting a proper number of random walks around the changed part. The newly started random walks help to adjust each node's times of visited by all random walk segments via adding or subtracting contribution from corresponding random walk segments. How IMCPR update PageRank scores when edges and nodes evolve are described respectively as follows.

3.1 IMCPR for Evolving Edges

Suppose an arbitrary edge $e(u, r)$ is added, we do M random walks starting from node r. For any node s, if s is passed through by any one of these random walks once, $X(s)$ is increased by one. M is a non-negative integer defined as Eq. (3). Then we do another M random walks. Each of these walks randomly picks one of u's outgoing neighbors except node r as its starting node. For any node s, if s is passed through by any one of these random walks once, $X(s)$ decreases by one.

Method for updating PageRank as edges removed is similar. Suppose an arbitrary edge $e(u, r)$ is removed, we do M random walks starting from node r. For any node s, if s is passed through by any one of these random walks once, $X(s)$ is decreased by one.

Then we do another M random walks. Each of these walks randomly picks one of u's outgoing neighbors except node r as its starting node. For any node s, if s is passed through by any one of these random walks once, $X(s)$ increases by one. Algorithm 1. presents the pseudo-code of updating PageRank for a graph with evolving edges.

$$M = \begin{cases} (1-c)X(u)/|N(u)|, & \text{edge } e(u,r) \text{ is added} \\ (1-c)X(u)/(|N(u)|-1), & \text{edge } e(u,r) \text{ is removed} \end{cases} \tag{3}$$

Algorithm 1 function ProcessingChangedEdges(*G*,*X*,*ΔE*,*c*,*R*)

Input: A directed graph *G* =(*V*,*E*), *ΔE* is set of edges newly added and/or removed, *c* is the reset probability, X is set of the number of random walk segments visiting each node initially, *R* is the number of random walks starting from each node in initial.

Output: π is the updated approximation of PageRank.

```
for each e(u,r) ∈ ΔE  do
    compute M according to Eq. (3)
    do M random walks starting from r
    for each random walk do
        if node s ∈ V is passed through by once do
            if e(u,r) is added do
                X(s)++
            else
                X(s)--
            end if
        end if
    end for
    for i = 1 to M do
        do 1 random walk starting from node v ∈ N(u) and
        v is not r
        if node s ∈ V is passed through by once do
            if e(u,r) is added do
                X(s)--
            else
                X(s)++
            end if
        end if
    end for
end for
π = cX/(|V|R)
return π
```

3.2 IMCPR for Evolving Nodes

Adding an arbitrary node r into graph G, can be regarded as two separate operations. First a node r is added, secondly edges associated with r are added. Similarly, an arbitrary node r removed means edges associated with r and the node r itself removed respectively. Thus, as an arbitrary node r added, we set $X(r)$ equal to R, times of random walks started from each node before, for initialization. Then we update PageRank scores according to the method described above. Supposes an arbitrary node r is removed, we set $X(r)$ equal to zero and update PageRank scores to process the removed edges. Adding the process for changed nodes and edges together, the pseudo-code of IMCPR algorithm is shown in Algorithm 2.

Algorithm 2 function IMCPR(*G*, *X*, *ΔG*, c, *R*)

Input: A directed graph *G* =(*V*, *E*), *ΔG* ={ *ΔE*, *ΔV*} denotes the newly changed edges and nodes, *c* is the reset probability, *X* is set of the number of random walk segments visiting each node initially, R is the number of random walks starting from each node in initial.

Output: π is the updated approximation of PageRank.

```
for each r ∈ ΔV and r is added do
        X(r) = R
end for
for each r ∈ ΔV and r is removed do
        X(r) = 0
end for
return ProcessingChangedEdges(G,X,ΔE,c,R)
```

4 Correctness and Time Complexity

4.1 Correctness Discussion

In this section, we prove that for an arbitrary node $u, \tilde{\pi}(u)$ got by IMCPR is sharply concentrated around its expectation, which is its real PageRank score $\pi(u)$.

Theorem 1. The expected PageRank score of an arbitrary node u got from IMCPR is equal to the real score. It is written as Eq. (4).

$$E[\tilde{\pi}(\mathrm{u})] = \pi(u) \tag{4}$$

Prove: It is proved that $E[\tilde{\pi}(\mathrm{u})]$ got by the original Monte Carlo based PageRank is equal to $\pi(u)$ in [4]. Thus, here we only need to prove that $E[\tilde{\pi}(\mathrm{u})]$ got by IMCPR is equal to the expectation got by the original algorithm.

For an arbitrary node u, IMCPR reuses $X(u)$ computed by the original Monte Carlo based PageRank in initial. It is equal to that IMCPR starting R random walks from each

node of the graph G. As an edge added or removed, IMCPR updates $X(u)$ as the method defined above, which is equal to rerouting the random walks which passes through the changed edge. The fundamental idea of IMCPR is that the expectation of contribution of $X(u)$ from an arbitrary edge $e(s, u)$ is equal to $(1-c)X(s)/|N(s)|$. Though the contribution of $X(u)$ from a particular edge varies, its expectation is steady.

The expected $X(u)$ for a node u in a Monte Carlo based algorithm can be computed as Eq. (5), where $P(v, u)$ donates the number of random walk segment which starts from v and visits u.

$$E[X(u)] = R \times E\left[\sum_{v \in V} P(v,u)\right] \tag{5}$$

The expected $P(v, u)$ for any node u and v only depend on the graph and how to choose next node in a random walk. As described in Sect. 3.1, it is straightforward that our proposed algorithm doesn't change the probability of choosing next node in random walks, so according to Eq. (5), we can tell that $E[X(u)]$ computed by IMCPR is equal to it computed by the original Monte Carlo based PageRank algorithm. Theorem 1 is proved.

Theorem 2. The PageRank score got from IMCPR is sharply concentrated around its expectation. Theorem 2 can be written as Eq. (6) for any node v, where δ is a concentrated factor and δ' is a constant depending on both δ and the reset probability c.

$$\Pr[|\tilde{\pi}(v) - \pi(v)| \geq \delta\pi(v)] \leq e^{-|V|R\pi(v)\delta'} \tag{6}$$

The proof of Theorem 2 is similar as some previous works [2, 10], but we still present the detailed derivation of the proof for the completeness of this paper.

Prove: Assuming $R = 1$(the situation that $R > 1$ can be proved like this), for an arbitrary node v, define $Z(u)$ donates c times of the visited time of node v got from the path start from node u. $Y(u)$ is the length of the random walk segment starting from u. Let $W(u) = cY(u)$, $z(u) = E[Z(u)]$. $Z(u)$ of different node u are independent. Thus,

$$\tilde{\pi}(v) = \sum\nolimits_{u \in V} Z(u)/|V|,\ \pi(v) = \sum\nolimits_{u \in V} z(u)/|V| \tag{7}$$

It is obvious that $0 \leq Z(u) \leq W(u)$ and $E[W(u)] = 1$.

From the definition of expectation, Eq. (8) can be derived.

$$\left(E\left[e^{hZ(u)}\right] - 1\right) / \left(E\left[e^{hW(u)}\right] - 1\right) \leq E[Z(u)]/E[W(u)] \tag{8}$$

$$E\left[e^{hZ(u)}\right] \leq z(u)E\left[e^{hW(u)}\right] + 1 - z(u) = z(u) \times \left(E\left[e^{hW(u)}\right] - 1\right) + 1 \tag{9}$$

Because for an arbitrary y meets $1+y \le e^y$, thus,

$$E\left[e^{hZ(u)}\right] \le e^{-z(u)\times\left(1-E\left[e^{hW(u)}\right]\right)} \tag{10}$$

$$\begin{aligned}\Pr[\tilde{\pi}(v) \ge (1+\delta)\pi(v)] &\le \frac{E\left[e^{h|V|\tilde{\pi}(v)}\right]}{e^{h|V|(1+\delta)\pi(v)}} = \frac{E\left[e^{h\sum_u Z(u)}\right]}{e^{h|V|(1+\delta)\pi(v)}} \le \frac{\prod_u E\left[e^{-z(u)\times\left(1-E\left[e^{hW(u)}\right]\right)}\right]}{e^{h|V|(1+\delta)\pi(v)}} \\ &= e^{-|V|\pi(v)\times\left(1-E\left[e^{hW(u)}\right]\right)} / e^{h|V|(1+\delta)\pi(v)} \le e^{-|V|\delta'\pi(v)}\end{aligned} \tag{11}$$

Similar as Eq. (11), Eq. (12) can be proved.

$$\Pr[\tilde{\pi}(v) \le (1-\delta)\pi(v)] \le e^{-|V|\delta'\pi(v)} \tag{12}$$

In addition, in Eqs. (11) and (12) $\delta' = 1 + h(1+\delta) - E\left[e^{hW}\right]$ where $W = cY$ is a random variable with Y having geometric distribution with parameter c. It means that the probability of the approximation deviated from its expectation is quite small and Theorem 2 is proved. So it is convinced that IMCPR performs as good as the original Monte Carlo based PageRank in accuracy. Above all, the correctness of IMCPR is proved.

4.2 Complexity Analysis

Supposing an arbitrary node r changed, IMCPR starts R random walks from r. Supposing an arbitrary edge $e(u, v)$ changed, there are $2 \times M$ random walks starting, including M random walks starting from node v and the same number of random walks starting from node u's other outgoing neighbors. Here we discuss the amount of operations as m nodes and n edges changed. The total number of newly started random walks in IMCPR, donated by *TotalRW*, can be calculated as Eq. (13).

$$Total\,RW = mR + 2\sum\nolimits_{e(u,v)\in\Delta E} M = mR + \sum\nolimits_{e(u,v)\in\Delta E} 2(1-c)X(u)/|N(u)| \tag{13}$$

For an arbitrary node u, its outgoing neighbors must be more than zero. So Eq. (14) must be true, where $\bar{X}(u)$ refers to the average number of random walk segments visiting node u in initial.

$$Total\,RW \le mR + 2\sum\nolimits_{e(u,v)\in\Delta E} X(u) = mR + 2n\bar{X}(u) \tag{14}$$

Supposing the edge $e(u, v)$ is inserted and/or removed randomly, $\bar{X}_t(u)$ can be calculated as Eq. (15).

$$\bar{X}(u) = \sum\nolimits_{u\in V} X(u)/|V| = R|V|/(c|V|) = R/c \tag{15}$$

The length of the newly started random walks is equal to the number of operation of update $X(u)$ for any arbitrary node u. So, we define the amount of computing operations of IMCPR donated by *TotalComp* as Eq. (16).

$$TotalComp = TotalRW/c \leq R/c(m + 2n/c) \tag{16}$$

The computational cost of IMCPR is only related to the size of changed parts of the graph. Algorithm in [2] takes complexity of $O(|V|\ln(n)/c^2)$ to update PageRank scores as n edges inserted and/or removed. We compare $|E|$ and $\ln(|E|)|V|$ for each graph from Stanford Network Analysis Project [11]. It is found that $|E|$ is closed to $\ln(|E|)|V|$ in most graphs and there are only 5 graphs which have $|E|$ obviously bigger than $\ln(|E|)|V|$. Meanwhile, some graphs such as memetracker, LiveJournal, wiki-Talk, web-Google and so on, their $|E|$ are much smaller than $\ln(|E|)|V|$. So we can tell that in many real-world applications with little percentages of graph changed, IMCPR takes a lower time complexity compared to algorithm in [2].

5 Experiments and Evaluations

5.1 Experimental Setup

We perform our experiments upon a five-machine homogeneous Hama [12] cluster. Each machine in the cluster has an intel-i7 2600 CPU, 2 GB memory, 4 TB hard disk and 1 Gigabit Ethernet card. Ubuntu 14.04 and zookeeper-3.4.6 are deployed on each machine. The version of Hama is hama-0.6.4 and HDFS component is provided by hadoop-1.2.1. Eight real-world graphs from widely used datasets [11] are used in our experiments. The key parameters of these graphs are listed in Table 1.

Table 1. Main parameters of data sets

Graph	p2p-Gnutella-31	Amazon-0312	Web-NotreDame	Web-BerkStan	Higgs-twitter	Wiki-talk	Wiki-vote	Email-Enron
Nodes	63K	401K	326K	685K	457K	2.3M	7.1K	37K
Edges	148K	3.2M	1.5M	7.6M	14.9M	5.0M	104K	184K
Dangling nodes	46K	12K	187K	4.7K	0.03K	2.2M	1K	0

In the experiments, we only consider the scenarios that edges and nodes are inserted into graphs, because removal is similar. In order to generate the edges inserted, we randomly choose 10% edges in each graph as evolving edges and use the rest part of each graph as initial graph. In order to generate the inserted nodes, we randomly add a certain percentage of nodes to each dataset, and appoint a stochastic incoming and outgoing neighbor for each newly added node. We set c = 0.15 and R = 20.

5.2 Comparison with Existing Algorithms

We evaluate the performance of our approach (Algorithm 2) by comparing it to the work in [2] and the non-incremental algorithm in [4]. For simplicity, we use BahmaniPR to refer to the algorithm proposed in [2] by Bahmani et al., and use BasicPR to refer to the original Monte Carlo based PageRank proposed in [4]. We also use PI to refer to Power Iteration [1] for short.

We evaluate the accuracy of the proposed algorithm with metric of $L1$ error which is defined as Eq. (17), where $\pi(u)$ is the "ground-truth" PageRank score of node u and $\tilde{\pi}(u)$ is the approximation. In experiments $\pi(u)$ is computed by PI algorithm (with parameter $\varepsilon = 5 \times 10^{-4}$).

$$Err = \sum\nolimits_{u \in V} |\tilde{\pi}(u) - \pi(u)| \tag{17}$$

We also evaluate the amount of computation of the proposed algorithm. We implement all the algorithms based on the BSP model [13] upon Hama. A message is sent as long as a random walker jumps to a node in these Monte Carlo based algorithms. So we get the amount of computation by counting the messages received by all machines (including messages received locally) during computation. We use Cost(Alg) to refer to the amount of computation of a particular algorithm Alg.

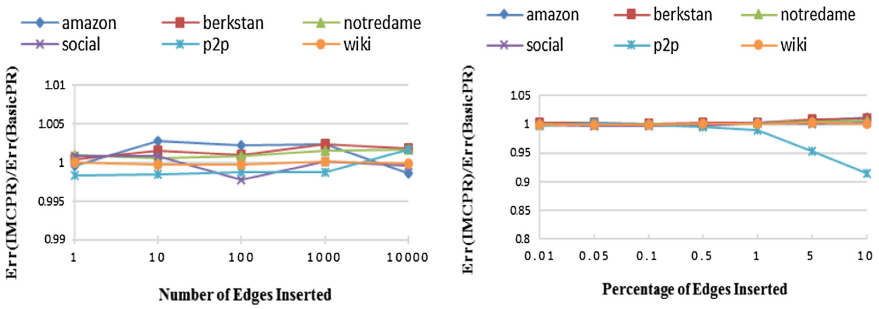
5.3 Accuracy

We evaluate the accuracy of the proposed algorithm. Firstly, we compare the average $L1$ error of BasicPR and IMCPR with 10% edges inserted. Table 2 describes the results. We found that the accuracy of IMCPR is roughly equal to original Monte Carlo based PageRank algorithm.

Table 2. Accuracy comparison of IMCPR and BasicPR with 10% edges inserted

Graph	Amazon0312	Web-BerkStan	Web-NotreDame	Higgs-twitter	p2p-Gnutella31	Wiki-talk
BasicPR	0.19	0.20	0.28	0.21	0.34	0.36
IMCPR	0.20	0.20	0.29	0.21	0.31	0.36

To verify that the errors do not accumulate as updating PageRank for evolving graph, we trace the accuracy of IMCPR as edges inserted continuously. There are p edges inserted respectively (p = 1, 10, 100, 1000, 1000). We record the errors of IMCPR and BasicPR. To make the figures intuitive, we depict Err(IMCPR)/Err (BasicPR) in the following figures. As Fig. 1(a) shows, the accuracy of IMCPR is stable and always close to the original Monte Carlo based PageRank algorithm. We also trace the accuracy as d percentages of edges inserted (d = 1%%,5%%,0.1%,0.5%,1%, 5%,10%). As Fig. 1(b) depicted, we found that the accuracy of IMCPR is always close to the baseline algorithm. In some cases (P2P with 5% and 10% edges inserted) IMCPR even gets slightly higher accuracy than the baseline algorithm does.



(a) 1 to 10000 edges inserted (b) 0.01% to 10% edges inserted

Fig. 1. Comparison of accuracy

5.4 Amount of Computation

Comparison with Existing Incremental Algorithm.
To demonstrate our proposed algorithm is efficient, we compare the amount of computation of our algorithm to BahmaniPR. We insert n edges (n = 50, 100, 150) in our experiments and record the amount of computation. We found that BahmaniPR cost 8.9 to 70 times as much amount of computation compared to our proposed algorithm which is depicted in Fig. 2.

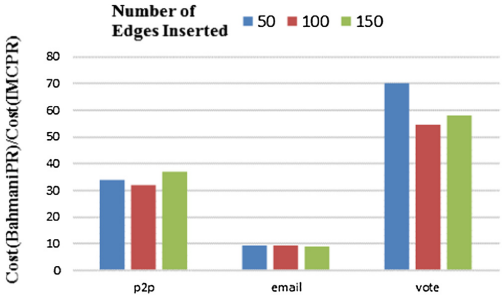
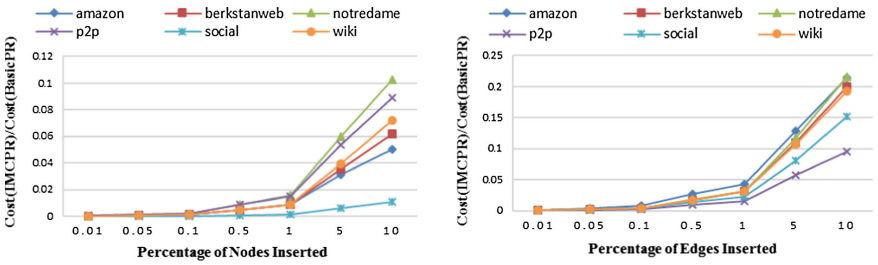


Fig. 2. Comparison of amount of computation to BahmaniPR

Comparison with Non Incremental Algorithm.
We also compare our proposed algorithm to original Monte Carlo based PageRank algorithm, we found our proposed algorithm reduces significant amount of computation compared to the original algorithm intuitively. Figure 3(a) and (b) describe the comparison of amount of computation of IMCPR and BasicPR in experiments of nodes and edges inserted respectively. As there is 1% data changed, IMCPR cuts down over 96% amount of computation at least. As 10% edges inserted IMCPR just cost 0.2 times amount of computation compared to the original algorithms at most.



(a) 0.01% to 10% nodes inserted (b) 0.01% to 10% edges inserted

Fig. 3. Comparison of amount of computation to BasicPR

Comparison as Different Number of Edges Changed.
Last but not least, we verifying the efficiency of IMCPR as different number of edges evolves. We compare the amount of computation of IMCPR as 1 to 10000 edges inserted mentioned above. We found that the amount of computation has a nearly linear correlation with the number of the edges inserted. These results consistent with the theoretical analysis in the previous section. Particular results are depicted in Fig. 4, to make the figure intuitive, we take the logarithms of amount of computation as the vertical axis.

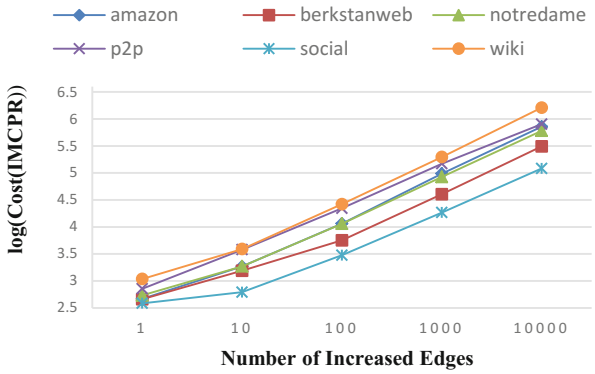


Fig. 4. Amount of computation of IMCPR as different numbers (1 to 10000) of edges inserted

6 Conclusion

In this paper, we investigate Monte Carlo based PageRank algorithms and propose an incremental algorithm called IMCPR, which significantly reduces the amount of computation for dynamic graphs. Both the theoretical analysis and experimental results with several typical real-world graphs demonstrate that IMCPR performs well in accuracy and performance. In addition, the proposed algorithm can be extended to Monte Carlo based Personalized PageRank [14], Single-Source Shortest Paths [15] and other random walk based algorithms.

Acknowledgement. We would like to thank Shan Shan for helpful suggestions.

References

1. Page, L., et al.: The PageRank citation ranking: bringing order to the web (1999)
2. Bahmani, B., Chowdhury, A., Goel, A.: Fast incremental and personalized pagerank. Proc. VLDB Endow. **4**(3), 173–184 (2010)
3. Desikan, P., et al.: Incremental page rank computation on evolving graphs. In: Special Interest Tracks and Posters of the 14th International Conference on World Wide Web. ACM (2005)
4. Avrachenkov, K., et al.: Monte Carlo methods in PageRank computation: when one iteration is sufficient. SIAM J. Numer. Anal. **45**(2), 890–904 (2007)
5. Langville, A.N., Meyer, C.D.: Deeper inside pagerank. Internet Math. **1**(3), 335–380 (2004)
6. Chien, S., et al.: Towards exploiting link evolution (2001)
7. Langville, A.N., Meyer. C.D.: Updating pagerank with iterative aggregation. In: Proceedings of the 13th International World Wide Web Conference on Alternate Track Papers & Posters. ACM (2004)
8. Kamvar, S., et al.: Exploiting the block structure of the web for computing pagerank. Technical report, Stanford University (2003)
9. Lofgren, P.: On the complexity of the Monte Carlo method for incremental PageRank. Inf. Process. Lett. **114**(3), 104–106 (2014)
10. Das Sarma, A., Molla, A.R., Pandurangan, G., Upfal, E.: Fast distributed PageRank computation. In: Frey, D., Raynal, M., Sarkar, S., Shyamasundar, Rudrapatna K., Sinha, P. (eds.) ICDCN 2013. LNCS, vol. 7730, pp. 11–26. Springer, Heidelberg (2013). doi:10.1007/978-3-642-35668-1_2
11. Jure, L.: Stanford Large Network Dataset Collection. http://snap.stanford.edu/data/index.html
12. Seo, S., et al.: HAMA: an efficient matrix computation with the mapreduce framework. In: 2010 IEEE Second International Conference on IEEE Cloud Computing Technology and Science (CloudCom) (2010)
13. Valiant, L.G.: A bridging model for parallel computation. Commun. ACM **33**(8), 103–111 (1990)
14. Jeh, G., Jennifer, W.: Scaling personalized web search. In: Proceedings of the 12th International Conference on World Wide Web. ACM (2003)
15. Pettie, S.: Single-source shortest paths. In: Encyclopedia of Algorithms, pp. 847–849 (2008)