Incremental Page Rank Computation on Evolving Graphs

Prasanna Desikan
Dept. of Computer Science
University of Minnesota
Minneapolis, MN 55455
USA
desikan@cs.umn.edu

Nishith Pathak
Dept. of Computer Science
University of Minnesota
Minneapolis, MN 55455
USA
npathak@cs.umn.edu

Jaideep Srivastava
Dept. of Computer Science
University of Minnesota
Minneapolis, MN 55455
USA
srivasta@cs.umn.edu

Vipin Kumar
Dept. of Computer Science
University of Minnesota
Minneapolis, MN 55455
USA
kumar@cs.umn.edu

ABSTRACT
Link Analysis has been a popular and widely used Web mining technique, especially in the area of Web search. Various ranking schemes based on link analysis have been proposed, of which the PageRank metric has gained the most popularity with the success of Google. Over the last few years, there has been significant work in improving the relevance model of PageRank to address issues such as personalization and topic relevance. In addition, a variety of ideas have been proposed to address the computational aspects of PageRank, both in terms of efficient I/O computations and matrix computations involved in computing the PageRank score. The key challenge has been to perform computation on very large Web graphs. In this paper, we propose a method to incrementally compute PageRank for a large graph that is evolving. We note that although the Web graph evolves over time, its rate of change is rather slow. When compared to its size. We exploit the underlying principle of first order markov model on which PageRank is based, to incrementally compute PageRank for the evolving Web graph. Our experimental results show significant speed up in computational cost, the computation involves only the (small) portion of Web graph that has undergone change. Our approach is quite general, and can be used to incrementally compute (on evolving graphs) any metric that satisfies the first order Markov property.

Keywords
Link Analysis, Web Search, PageRank, Incremental Algorithms

1. INTRODUCTION
The importance of link analysis on the Web graph has gained significant prominence after the advent of Google [1]. The key observation is that a hyperlink from a source page to a destination page serves as an endorsement of the destination page by the (author of the) source page on some topic. This idea has been exploited by various researchers and has resulted in a variety of hyperlink based ranking metrics for ranking of Web Pages. Kleinberg's Hubs and Authority [2] and Google's Pagerank [3] are the most popular among such metrics. A variety of modifications and improvements to these approaches have been developed in recent years[6,7,8,9,10].

Link analysis techniques have adopted different knowledge models for the measures developed for various applications on the Web [15]. Kleinberg's Hubs and Authority is based on the observation that the Web graph has a number of bipartite cores [2], while Google's PageRank is based on the observation that a user's browsing of the Web can be approximated as a first order markov model [3]. Giles, et al [5] have used network flow models to identify web communities. Thus, a variety of models have been used to measure different properties of the Web Graph at a given time instance. Success of Google has signified the importance of Pagerank as a ranking metric. This has also led to a variety of modifications and improvisations of the basic PageRank metric. These have either focused on changing the underlying model or on reducing the computation cost.

Another important dimension of Web mining is the evolution of the Web graph [4]. The Web is changing over time, and so is the users' interaction on (and with) the Web, suggesting the need to study and develop models for the evolving Web Content, Web Structure and Web Usage. The study of such evolution of the Web would require computing the various existing measures for the Web graph at different time instances. A straightforward approach would be to compute these measures for the whole Web Graph at each time instance. However, given the size of the Web graph, this is becoming increasingly infeasible. Furthermore,, if the percent of nodes that change during a typical time interval when the Web is crawled by search engines is not high, a large portion of the computation cost may be wasted on re-computing the scores for the unchanged portion. Hence, there is a need for computing metrics incrementally, to save on the computation costs.

Techniques for incremental computations, to study changes in graph structure over time, would depend on the underlying knowledge model that defines a metric [15]. For example, the computation of hub and authority scores is based on mutual reinforcement of nodes, and hence a change in the indegree or outdegree of a node may affect its score. Mutual reinforcement makes hub and authority scores a second order model. However, for PageRank whose random surfer model is based on the first order markov property, the change in out degree of the node does not affect the score of the node. Hence, the level of penetration of change in scores due to a change in the degree of a node is not as high in PageRank as in hub and authority scores.

In this paper, we describe an approach to compute PageRank in an incremental fashion. We exploit the underlying first order markov model[1] property of the metric, to partition the graph

WWW 2005, May 10--14, 2005, Chiba, Japan.

[1] The property that the PageRank score of a page depends only on the PageRank scores of the pages pointing to it.

into two portions such one of them is unchanged since the last computation, and it has only outgoing edges to the other partition. Since there are no coming edges from the other partition, the distribution of PageRank values of the nodes in this partition will not be affected by the nodes in the other partition. The other partition is the rest of the graph, which has undergone changes since the last time the metric was computed. Figure 1 gives an overview of our approach and explains the difference between related work and the work in this paper. This paper is organized as follows. In Section 2, we give an introduction to the basic PageRank metric and the various issues involved in its computation. Section 3 gives an overview of our approach to incrementally compute PageRank for evolving Web graphs. We describe the Incremental PageRank Algorithm in Section 4 and present our experimental results in Section 6. Section 7 discusses the related work and places our work in context. Finally, in Section 8 we conclude the and provide directions for future work.

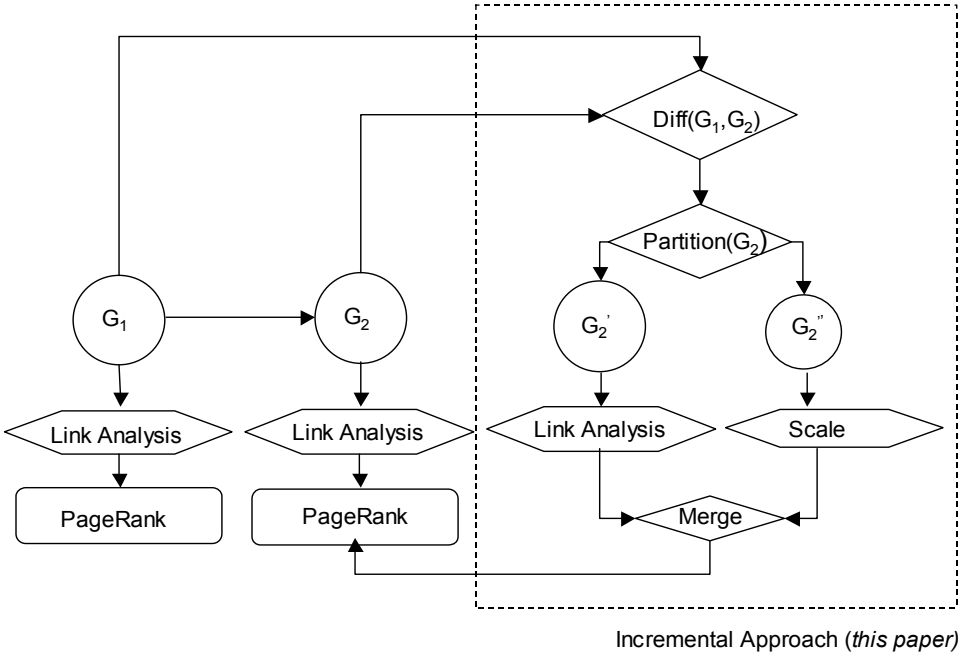


Figure 1. Overview of the Proposed Approach.

2. PAGERANK

PageRank is a metric for ranking hypertext documents that determines their quality. It was originally developed by Page et al. [3] for the popular search engine, Google [1]. The key idea is that a page has high rank if it is pointed to by many highly ranked pages. Thus, the rank of a page depends upon the ranks of the pages pointing to it. The rank of a page p can thus be written as:

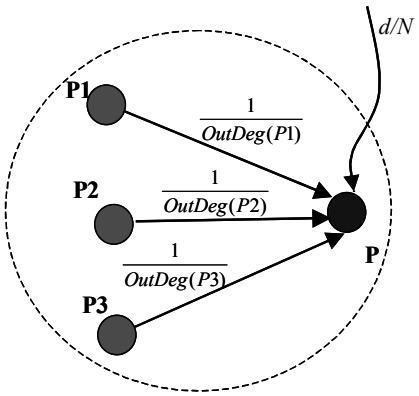
$$PR(p) = \frac{d}{n} + (1-d) \cdot \sum_{(q,p) \in G} \frac{PR(q)}{OutDegree(q)} \qquad (1)$$

Here, n is the number of nodes in the graph and OutDegree(q) is the number of hyperlinks on page q. Intuitively, the approach can be viewed as a stochastic analysis of a random walk on the Web graph. The first term in the right hand side of the equation corresponds to the probability that a random Web surfer arrives at a page p from somewhere, i.e. (s)he could arrive at the page by typing the URL or from a bookmark, or may have a particular page as his/her homepage. d would then be the probability that a random surfer chooses a URL directly – i.e. typing it, using the bookmark list, or by default – rather than traversing a link. Finally, 1/n is the uniform probability that a person chooses page p from the complete set of n pages on the Web. The second term in the right hand side of the equation corresponds to a factor contributed by arriving at a page by traversing a link. 1- d is the probability that a person arrives at the page p by traversing a link. The summation corresponds to the sum of the rank contributions made by all the pages that point to the page p. The rank contribution is the PageRank of the page multiplied by the probability that a particular link on the page is traversed. So for any page q pointing to page p, the probability that the link pointing to page p is traversed would be 1/OutDegree(q), assuming all links on the page is chosen with uniform probability. Figure 2 illustrates an example of computing PageRank of a page P from the pages, P1, P2, P3 pointing to it.

There are other computational challenges that arise in PageRank. Apart from the issue of scalability, the other important computational issues are the convergence of PageRank iteration and the handling of dangling nodes. The convergence of PageRank is guaranteed only if the Web graph is strongly connected and is aperiodic. To ensure the condition of strong connectedness, the dampening factor is introduced, which assigns a uniform probability to jumping to any page. In a graph theoretic sense it is equivalent of adding an edge between every pair of vertices with a transition probability of d/n. The aperiodic property is also guaranteed for the Web graph.

Another important issue in computation of PageRank is the handling of dangling nodes. Dangling nodes are nodes with no outgoing edge. These nodes tend to act as rank sink, as there is no way for rank to be distributed among the other nodes. The suggestion made initially to address this problem, was to iteratively remove all the nodes that have an outdegree of zero,

and compute the PageRank on the remaining nodes [3]. The reasoning here was that dangling nodes do not affect the PageRank of other nodes. Another suggested approach was to remove the dangling nodes while computation initially and add them back during the final iterations of the computation [7]. Other popular approaches to handling dangling nodes, is to add self loops to dangling nodes[11,20] and to add links to all nodes in the graph, G from each of the dangling node to distribute the PageRank of the dangling node uniformly among all nodes[3].



$$PR(P) = d/N + (1-d)\left(\frac{PR(P1)}{OutDeg(P1)} + \frac{PR(P2)}{OutDeg(P2)} + \frac{PR(P3)}{OutDeg(P3)}\right)$$

Figure 2. Illustrative example of PageRank.

3. PROPOSED APPROACH

In the proposed approach, we exploit the underlying first order Markov Model on which the computation of PageRank is based. It should be noted that PageRank of a page depends only on the pages that point to it and is independent of the outdegree of the page. The principle idea of our approach is to find a partition such that there are no incoming links from a partition, Q (includes all changed nodes) to a partition, P. In such a case the PageRank of the partition, Q is computed separately and later scaled and merged with the rest of the graph to get the actual PageRanks of vertices in Q. The scaling is done with respect to the number of vertices in partition, P-`n(P) to the total number of nodes in the whole graph, G –n(P UQ)=V. The PageRank of the partition Q is computed, taking the border vertices that belong to the partition P and have edges pointing to the vertices in partition Q. The PageRank values of partition P are obtained by simple scaling.

This basic idea of partitioning the Web graph, and computing the PageRanks for individual partitions and merging works extremely well when incrementally computing PageRank for a Web graph that has evolved over time. Given, the Web graphs at two consecutive time instances, we first determine the portion of the graph that has changed. A vertex is declared to be changed when a new edge added or deleted between the vertex and any other vertex belonging to the graph or if the weight of a node or an edge weight adjoined to that node has changed. Once the changed portion is defined, for each page we determine iteratively all the pages that will be affected by its PageRank. In this process, we include pages that remain unchanged but whose; PageRank gets affected due to the pages that have changed. in partition Q. The rest of the unchanged graph is in partition P.

The whole concept is illustrated in Figure 3. Let the graph at the new time be G(V,E), and

v_b = Vertex on the border of the left partition from which there are only outgoing edges to the right partition.

v_{ul} = vertex on the left partition which remains unchanged

The set of unchanged vertices can be represented as,

$V_u = \{v_u, \forall v_u \in V\}$ where v_u is a vertex which has not changed.

v_{ur} = Vertex on the right partition which remains unchanged, but whose PageRank is affected by vertices in the changed component.

v_{cr} = Vertex on the right partition which has changed, or has been a new addition.

Therefore, the set of changed vertices can be represented as,

$$V_c = \{v_{cr}, \forall v_{cr} \in V\}$$

In order to compute PageRank incrementally, for every vertex in V_c, which is a set of changed vertices, perform a BFS to find out all vertices reachable from this set. The PageRank of these vertices will be affected by vertices in V_{cr}. These set of vertices can be denoted by the set,

$$V_{ur} = \{v_{ur}, \forall v_{ur} \in V\}$$

Similarly, the set of vertices v_b can be denoted as,

$$V_b = \{v_b, \forall v_b \in V\}$$

Hence the set of vertices whose PageRank has to be computed in the incremental approach corresponds to the partition Q described above, and can be denoted as,

$$V_Q = V_c \cup V_{ur} \cup V_b$$

Let an edge set, E_Q, be defined as set of edges,

$E_Q = \{e_{x,y} \mid x, y \in V_Q\}$, where $e_{x,y}$ represents a directed edge from vertex *x* to vertex *y*.

The set of partitioning edges can be defined as,

$$E_{Part} = \{e_{x,y} \mid x \in V_P, y \in V_Q\},$$

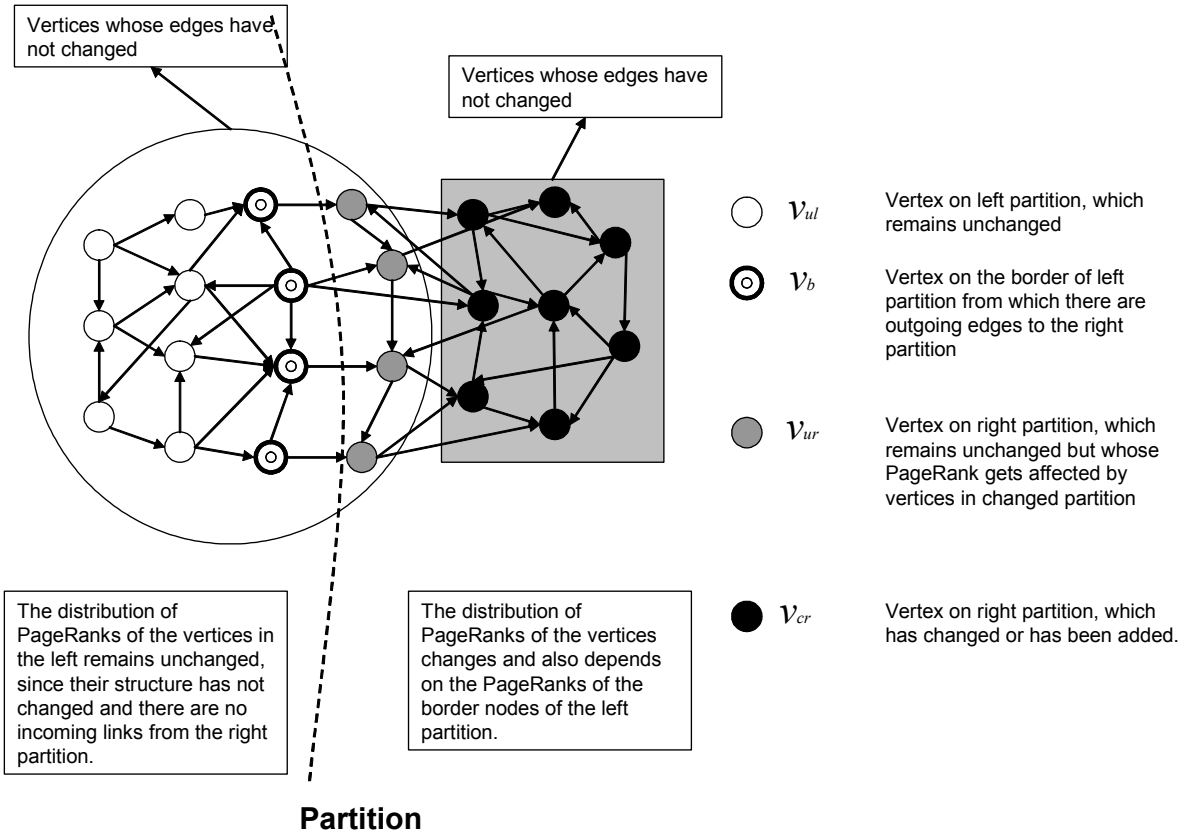


Figure 3. Incremental Computation of PageRank.

The vertices in partition P can be defined as,

$$V_P = V - V_Q + V_b$$

And the edges that correspond to this partition can be defined as,

$E_P = \{e_{x,y} \mid x, y \in V_P\}$, where $e_{x,y}$ represents a directed edge from vertex *x* to vertex *y*.

Thus, the given graph $G(V, E)$ can be partitioned into two graphs namely, $G_P(V_P, E_P)$ and $G_Q(V_Q, E_Q)$.

Now, since we know that the graph $G_P(V_P, E_P)$ has remained unchanged from the previous time instance and the PageRank of vertices in this partition is not affected by the partition, $G_Q(V_Q, E_Q)$. Now a change in a node induces a change in the distribution of PageRank values for all its children and since all the nodes that are influenced by changes are already separated in the partition Q. The distribution of PageRank values for the nodes in partition G_P is going to be the same as it was for the corresponding nodes in the previous time instance G'. Thus the PageRank of the vertices in partition *P* could be calculated by simply scaling the scores from the previous time instance. And the scaling factor will be $n(G')/n(G)$, where G' is the graph at the previous time instance. And the PageRank for the partition, $G_Q(V_Q, E_Q)$ can be computed using the regular PageRank Algorithm and scaled for the size of the graph, G. Since the percent change in the structure of the Web is not high, the computation of the changed portion will be a smaller graph compared to the whole Web. And the existence of such partitions is also suggested by the bow-tie model of the Web [12], where about 27% of Web contributes to the influx. It should also be noted that while computing PageRanks for the changed portion, in order to maintain the stochastic property of the incremental matrix, we have to scale the PageRanks of nodes in V_b such that they correspond to the number of nodes for which the PageRank is actually computed. Also taken into account is the outdegree of these border nodes that have edges in partition P, since the way they distribute their PageRanks to nodes in partition Q, will depend on their outdegree.

4. INCREMENTAL ALORITHM

In this section, we will describe the incremental algorithm to compute PageRank. The initial step is to read the graph at a new instance and determine the vertices that have changed. This does not require additional time as it can be computed as we read the new graph. Thus, after reading the graph, we can assume that we are given two sets of vertices – one containing the vertices which

have changed from a previous time instance and the other containing vertices that have remain unchanged. Hence, the input to the algorithm is the graph G, and the two lists V_c and V_u. The outline of the algorithm is shown in Figure 3. We will describe each step briefly:

Step 1 - Initialize a list V_Q

Step 2 - A change in a vertex induces a change in the PageRank distribution of all its children. All such changed vertices are in the queue V_c. In this step, the list of "changed vertices" is extended to a partition to include all descendents of the initial list of "changed vertices". All these vertices are pushed into the list Q2.

Step 3 – For the remaining vertices are there is no change in their PageRank distribution. The New PageRank is simply obtained by scaling the previous PageRank scores. The scaling factor is simply:

$n(G')/n(G)$=Order of graph at previous time instance/Order of the graph at the present time instance.

Also all those vertices from this set of unchanged vertices that point to a changed vertex, will influence the PageRank value of that changed vertex, hence these too must be included in the list V_Q as their PageRank scores will be required for computing the PageRank scores for the changed vertices.

Step 4 – Now original PageRank computation algorithm along with steps taken to ensure stochastic property of transition matrix is performed on the nodes that are in Q2 and colored violet (i.e. nodes which have changed) to get the new PageRank values for these changed nodes.

Thus, we end up localizing the changed partition to a certain sub-graph of the web which consists of all changed nodes and then basic PageRank algorithm is performed only on this changed sub-graph. The PageRank value for the rest of the nodes is simply a matter of scaling the previous values.

IPR(G,V_u,V_c) :-
Step 1 – Initialize the list V_Q
Step 2 – Pop a Vertex N from V_c
2.1 For all the children of N
if children of N∈ list V_u
remove them from V_u
push them in V_c
2.2. Push N in V_Q and repeat step 2 till
queue V_c is empty
Step 3 – For each element in list V_u
3.1 Take the element and scale the previous pagerank value to get new pagerank value.
3.2 Look up whether any of the children, of the element of V_u belong to V_Q, if so remove this element of V_u, copy it in V_b .
Step 4 – Scale Border Nodes in V_b for stochastic property
Perform Original PageRank($V_Q \cup V_b$)

Figure 4. Incremental PageRank Algorithm.

Step 2 has a cost of E', where $E' = E_Q - E_{Part}$, is the number of edges in the partition Q. Now the PageRank values for the partition P are obtained by scaling the PageRank values with respect to ranks in the previous time instance. This step requires a cost of V'', where V'' is number of vertices in partition P. Now using these scaled values and the naïve approach PageRank for the vertices in partition Q is calculated. This step (including that required to scale the border nodes) requires a cost of $nE + E' + V_b$, where n is number of iterations required for PageRank values to converge and E' is again number of edges in partition Q. Thus, the total cost for incremental PageRank can be summed up to be $O(2E'+V''+nE+V_b)$.

5. EXPERIMENTAL RESULTS

To test our theoretical approach on real datasets, we needed graphs at two different time instances to compute the incremental version. We performed the experiments on two different web sites- the Computer Science website and the Institute of Technology website at the University of Minnesota. We performed the experiments at different time intervals to study the change and effect of the incremental computation. For the Computer Science website our analysis was done at a time interval of two days, eight days and ten days. We also performed the analysis for a time interval of two days for the Institute of technology web site.

In our experiments we also simulated the focused crawling, by not considering the Web pages that have very low PageRank into our graph construction and PageRank Computation. This was to emulate the real world scenario where not all pages are crawled. We wanted to analyze, how the incremental approach performs when pages with low PageRank are not crawled.

We used the following approximate measure to compare the computational costs of our method versus the naïve method.

*Number of Times Faster = Num of Iterations(*PR*)/(*1 + *(fraction of changed portion)*Number of iterations(*IPR*))*

The intuition behind the measure was how fast the convergence threshold will be reached computing PageRank incrementally versus computing PageRank in a naïve method for the whole graph. The convergence threshold that was chosen on our experiments was $1x10^{-8}$

The experimental results are presented in Figure 5. These results are from actual experiments conducted on the Computer Science and Institute of Technology websites. For the Computer Science website, in the first time interval of eight days, there seemed to be a significant change in the structure of the Website – about 60% of the pages had changed their link structure. We found out such a sea change occurred because a whole subgraph that contained the documentation for Matlab help was removed. The incremental approach still however, performed 1.86 as much faster as the naïve PageRank. Similarly, for a period of ten days the incremental approach performed around 1.75 times faster. For a period of two days the improvement was 8.65 times faster. These results are for the case of an unfocussed crawl. The results for focused crawl for the CS Website were better. In the first case, when the time interval was eight days, the improvement was 1.9 times and when the time interval was 10 days, the improvement was 1.76 times. For a period of two days the improvement with

focused crawling was 9.88 times. Thus, it suggests that focused crawling can also improve the computational costs of the incremental algorithm.

Computer Science Website

Focussed Crawl			
July19 vs July 27th			
percentage of change = 53.1429%	L1 -norm : 4.38609e-05	**NumTimes faster=**	**1.900538**
10 iteration(s) for inc_pagerank			
12 iteration(s) for actual pagerank			
July 27th vs July 29th			
percentage of change = 5.25071%	L1-norm : 1.60988e-07	**NumTimes faster=**	**9.885481**
6 iteration(s) for inc_pagerank			
13 iteration(s) for actual pagerank			
July19th vs 29th			
percentage of change = 58.3493%	L1-norm : 4.38692e-05	**NumTimes faster=**	**1.755669**
10 iteration(s) for inc_pagerank			
12 iteration(s) for actual pagerank			

Unfocussed Crawl			
July19 vs July 27th			
percentage of change = 60.2997%	norm : 1.70552e-07	**NumTimes faster=**	**1.867123**
9 iteration(s) for inc_pagerank			
12 iteration(s) for actual pagerank			
July 27th vs July 29th			
percentage of change = 5.56966%	norm : 1.51747e-07	**NumTimes faster=**	**8.659162**
9 iteration(s) for inc_pagerank			
13 iteration(s) for actual pagerank			
July 19th vs July 29th			
percentage of change = 65.0586%	norm : 1.60377e-07	**NumTimes faster=**	**1.749526**
9 iteration(s) for inc_pagerank			
12 iteration(s) for actual pagerank			

(a)

Institute of Technology Website

Unfocussed/Focussed Crawl			
July 30th vs Aug 1st			
percentage of change = 0%	norm : 8.15708e-07	**NumTimes faster=**	**11**
0 iteration(s) for inc_pagerank			
11 iteration(s) for actual pagerank			

(b)

Figure 5. Comparison of results for Incremental PageRank Algorithm versus Naïve PageRank Algorithm for the following departments at the University: (a) Computer Science Website, (b) Institute of Technology Website**.**

The Institute of technology website typically represented a website that doesn't change too often. The change over a period of two days in the Web Structure was none. Since there was no change detected, there was no necessity to compute the PageRank for the graph at the new time instance. And by our measure, it was 11 times faster. Since, there was no change in the graph structure, the improvements for the case of focused crawling and unfocussed crawling remain the same.

6. RELATED WORK

Determining the quality of a page has been the primary focus of Web mining research community and various measures and metrics have been developed for the same for different applications. PageRank [3] was developed by Google founders, for ranking hypertext documents. The overall idea is described in detail in Section 2. The other popular metric based on link analysis is hub and authority scores. From a graph theoretic point

of view, hubs and authorities can be interpreted as 'fans' and 'centers' in a bipartite core of a Web graph. The hub and authority scores computed for each Web page indicate the extent to which the Web page serves as a 'hub' pointing to good 'authority' pages or the extent to which the Web page serves as an 'authority' on a topic pointed to by good hubs. The hub and authority scores for a page are not based on a formula for a single page, but are computed for a set of pages related to a topic using an iterative procedure, namely HITS algorithm [2]. A detail study of link analysis techniques can be found in [13, 14, 15, 21, 22].

There have been a number of extensions of the basic PageRank that have been proposed, such as including the topic information of page to determine the topic relevance. One approach [17] was to precalculate different PageRank vectors for a given number of terms, focusing on the subset of pages that contain the term of interest. The search results for a query would be ranked according to scores that were precalculated for the collection of terms that contain the query words. Another approach for introducing topic relevance was addressed by Haveliwala et al [9]. In the approach, PageRank is calculated for all pages according to each category of the Open directory project. The pages that belong to a particular category have higher scores for the PageRank values computed for that category. Ranking of results of a search query is done according to scores of the category in which the query terms belong to. Oztekin et al [16], proposed Usage Aware PageRank. Their modified PageRank metric incorporates usage information. Weights are assigned to a link based on number of traversals on that link, and thus modifying the probability that a user traverses a particular link. Also the probability to arrive at a page directly is computed using the usage statistics.

There has been a variety of work on improving on the PageRank computation. I/O efficient techniques for computing PageRank has been addressed by Haveliwala et al [8] and Yen Yu Chen[10]. The basic of their approach is to partition the link file of the whole graph into partitions such that destination vector of each partition fits into the main memory. Kamvar et al in [6] have suggested quadratic extrapolation techniques to accelerate the convergence of PageRank. In a different paper [7], they have also suggested a way of exploiting the block-structure of the Web to compute *Block Ranks* for different domains and compute local PageRanks. Chein et al [19] have also exploited the idea of evolving graph to compute PageRank. However, their idea is to collapse the unchanged portion to a single node and compute the PageRank for the new graph. This leads to approximate PageRank values.

In out paper, we provide an approach to incrementally compute PageRanks. We do so by exploiting the underlying first order Markov model on which PageRank is based and partition the graph in such a manner so that we compute the exact PageRank values for a graph at a new time instance. Incremental computations are very useful to study the evolution of graphs. The significance of to study the temporal behavior of graph is addressed in our earlier paper [4].

7. CONCLUSIONS AND FUTURE DIRECTIONS

In this paper we have provided an approach to compute PageRank incrementally for evolving graphs. The key observation is that evolution of the Web graph is slow, with large parts of it remaining unchanged. By carefully delineating the changed and unchanged portions, and the dependence across them, it is possible to develop efficient algorithms for computing the PageRank metric incrementally. We follow a generic approach that can be applied to any algorithm that has been developed for efficient computation of the PageRank metric. Experimental results show significant speed up in computation of PageRank using our approach as compared to naive approach. Also, in the incremental approach, if the partitioned sub-graph that has changed is small, the whole PageRank computation might perhaps be performed in main memory.

Many issues remain to be investigated. In this paper we have proposed an incremental approach that applies to graph metrics based on first order Markov model, such as PageRank. An area to explore is for similar incremental approaches for other link based metrics. We have provided a method for an efficient incremental computation of relevance metric for a single node level. However, to study graph evolution, we would need measures and metrics defined at the level of a subgraph and a whole graph, and efficient methods to incrementally compute them

8. ACKNOWLEDGMENTS

We would like to thank Data Mining Research group at the Department of Computer Science for providing valuable suggestions. This work was been partially supported by the ARDA Agency under contract F30602-03-C-0243 and Army High Performance Computing Research Center contract number DAAD19-01-2-0014. The content of the work does not necessarily reflect the position or policy of the government and no official endorsement should be inferred. Access to computing facilities was provided by the AHPCRC and the Minnesota Supercomputing Institute.

9. REFERENCES

[1] http://www.google.com

[2] J.M. Kleinberg, "Authoritative Sources in Hyperlinked Environment", 9th Annual ACM-SIAM Symposium on Discrete Algorithms, pages 668-667, 1998

[3] L. Page, S. Brin, R. Motwani and T. Winograd "The PageRank Citation Ranking: Bringing Order to the Web" Stanford Digital Library Technologies, January 1998.

[4] P. Desikan and J. Srivastava, "Mining Temporally Evolving Graphs", WebKDD Workshop, Seattle (2004).

[5] Gary William Flake, Steve Lawrence, C. Lee Giles . *Efficient Identification of Web Communities***.** Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. August 2000, pp. 150-160.

[6] Sepandar D. Kamvar, Taher H. Haveliwala, Christopher D. Manning, and Gene H. Golub, "Extrapolation Methods for Accelerating PageRank Computations." In *Proceedings of the Twelfth*

International World Wide Web Conference, May, 2003.

[7] Sepandar D. Kamvar, Taher H. Haveliwala, Christopher D. Manning, and Gene H. Golub, "Exploiting the Block Structure of the Web for Computing PageRank." *Preprint*, March, 2003

[8] Taher Haveliwala. "Efficient Computation of PageRank," Stanford University Technical Report, September 1999.

[9] Taher Haveliwala. "Topic-Sensitive PageRank," In Proceedings of the Eleventh International World Wide Web Conference, May 2002

[10] Y. Chen, Q. Gan, and T. Suel. I/O-efficient techniques for computing pagerank. In Proc. of the 11th International Conf. on Information and Knowledge Management, pages 549--557, November 2002.

[11] G. Jeh and J. Widom. Scaling personalized web search. In 12th Int. World Wide Web Conference, 2003.

[12] R. Kumar, P. Raghavan, S. Rajagopalan, D. Sivakumar, A. Tomkins, and E. Upfal. The Web as a graph. In ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, pages 1-10, 2000.

[13] Kemal Efe, Vijay Raghavan, C. Henry Chu, Adrienne L. Broadwater, Levent Bolelli, Seyda Ertekin (2000), The Shape of the Web and Its Implications for Searching the Web, International Conference on Advances in Infrastructure for Electronic Business, Science, and Education on the Internet- Proceedings at http://www.ssgrr.it/en/ssgrr2000/proceedings.htm, Rome. Italy, Jul.-Aug. 2000.

[14] Monika Henzinger, *Link Analysis in Web Information Retrieval,* ICDE Bulletin Sept 2000, Vol 23. No.3.

[15] P. Desikan, J. Srivastava, V. Kumar, P.-N. Tan, "Hyperlink Analysis – Techniques & Applications", Army High Performance Computing Center Technical Report, 2002.

[16] B.U. Oztekin, L. Ertoz and V. Kumar, "Usage Aware PageRank", World Wide Web Conference, 2003.

[17] M. Richardson and P. Domingos. The intelligent surfer: Probabilistic combination of link and content information in pagerank. In Advances in Neural Information Processing Systems, 2002.

[18] J. Srivastava, P.Desikan and V.Kumar. "Web Mining - Concepts, Applications and Research Directions." Book Chapter in Data Mining: Next Generation Challenges and Future Directions, MIT/AAAI 2004.

[19] S. Chien, C. Dwork, S. Kumar, and D. Sivakumar. Towards exploiting link evolution. Unpublished manuscript, 2001.

[20] Eiron, N., McCurley, K., Tomlin, J.: Ranking the Web frontier. In: Proc. 13th conference on World Wide Web, ACM Press (2004) 309—318

[21] S.Acharyya and J.Ghosh, "A Maximum Entropy Framework for Link Analysis on Directed Graphs", in LinkKDD2003, pp 3-13, Washington DC, USA, 2003

[22] C. Ding, H. Zha, X. He, P. Husbands and H.D. Simon, "Link Analysis: Hubs and Authorities on the World Wide Web" May 2001. LBNL Tech Report 47847.

10. APPENDIX: PROOF OF ALGORITHM'S SCALABILITY

Let order of graph G be n

Let weight of a node v_i be w_i, $\forall\, v_i \in V$

Also, $\sum_{i=1}^{n} w_i = W$, sum of the weights of all nodes.

PageRank score calculation is analogous to the convergence of a first order Markov chain.

For calculation of PageRank we perform the operation,

$$TM_i = M_{i+1}$$

over a number of iterations. Here, T is the transition matrix and M_i is the PageRank score vector at the end of ith iteration.

We also have initial PageRank score vector,

$$M_0 = \begin{bmatrix} w_1/W \\ w_2/W \\ . \\ . \\ . \\ . \\ . \\ w_n/W \end{bmatrix}$$

For a node s we have,

$$PR(s) = d_f(w_s/W) \quad + \quad (1-d_f)\sum_{i1=1}^{k1} PR(x_{i1}) \Big/ Out\deg ree(x_{i1})$$

where, d_f is the dampening factor.

$x_{i1}\ \forall\, i1 = 1,2,...,k1$ are all those nodes that have at least one outgoing edge to s

For the sake of representation let us assume that all those nodes that have outgoing edge(s) to a node x_{im} are represented as $x_{i(m+1)}$.

Now, if l iterations are required for convergence towards the PageRank score vector then we have,

$$PR(s) = d_f(w_s/W) \quad + \quad (1-d_f)\sum_{i1=1}^{k1} PR(x_{i1}) \Big/ Out\deg ree(x_{i1})$$

$$PR(s) = d_f(w_s/W) + (1-d_f)\sum_{i1=1}^{k1} \frac{\{d_f(w_{x_{i1}}/W) + (1-d_f)\sum_{i2=1}^{k2}\{..\sum_{il=1}^{kl}(w_{x_{il}}/W \cdot Out\deg ree(x_{il}))..\}/Outdergee(x_{i2})\}}{Out\deg ree(x_{i1})}$$

Notice that the term W can be taken out as common. Thus, we have,

$$PR(s) = \frac{1}{W}[d_f + (1-d_f)\sum_{i1=1}^{k1} \frac{\{d_f w_{x_{i1}} + (1-d_f)\sum_{i2=1}^{k2}\{..\sum_{il=1}^{kl}(w_{x_{il}}/Out\deg ree(x_{il}))..\}/Out\deg ree(x_{i2})\}}{Out\deg ree(x_{i1})}]$$

$$\therefore PR(s) = \frac{\alpha}{W}$$

Consider a Graph $G(V,E)$

where,

$$\alpha = [d_f + (1-d_f)\sum_{i1=1}^{k1} \frac{\{d_f w_{x_{i1}} + (1-d_f)\sum_{i2=1}^{k2}\{..\sum_{il=1}^{kl}(w_{x_{il}}/Out\deg ree(x_{il}))..\}/Out\deg ree(x_{i2})\}}{Out\deg ree(x_{i1})}]$$

Now, suppose graph $G(V,E)$ changes to $G'(V',E')$

However the changes that occur are such that there is no change in the structure and weight of the of the node s as well as no change in the in the structure and weight of all the ancestors of the node s ..... [**condition (1)**].

Now following along the lines of the previous graph we have, for the new graph G'

$$PR'(s) = \frac{\alpha'}{W'}$$

The terms α and α' depend mainly only on the structure of the graph and the weights of, node s and its ancestors, from **condition(1)** we can see that both of them are identical in the graphs G and G' for the node s, assuming that the same dampening factor is used for both the pagerank calculations.

Thus, $\alpha = \alpha'$

$$\therefore \quad W \cdot PR(s) = W' \cdot PR'(s)$$

$$\Rightarrow \quad PR'(s) = (\frac{W}{W'})PR(s) \text{ .....} \textbf{result(1)}$$

Thus, the new pagerank score of any node x in G' is simply obtained by scaling the pagerank of score of the same node x in G by a factor of (W/W') provided that **condition(1)** holds for the node x and the same dampening factors are used for both pagerank calculations.

In most cases all nodes of a graph are equally likely here we have,

$$w_1 = w_2 = w_3 = ... = w_n = 1$$

$$\therefore \quad W = n(G),$$

where $n(G)$ is the order of the graph G.

When this graph changes to a graph G' we have **result(1)**; in this case as,

$$PR'(x) = \frac{n(G)}{n(G')} PR(x)$$

From the above result it is also trivial to deduce that a change in a node influences the pagerank values of all its descendents.