DYNA-RANK: Efficient calculation and updation of PageRank

Mandar Kale
National Institute of Technology Karnataka
India 575025
mandarakale@gmail.com

Mrs. P.Santhi Thilagam
National Institute of Technology Karnataka
India 575025
santhi_soci@yahoo.co.in

Abstract

The decision of the ranking of web page is very important in web, as its growing and changing very rapidly. Ranking of the results in a search engine for a query plays crucial role for huge database like Web, where one query can have millions of results. The browsing nature of web will mostly depend on the ranking of the search results. The existing approaches for calculating pagerank values are mostly centralized and the ones which are distributed, are not being used for practical purposes because of the scalability reasons. The centralized approaches considers total web as one graph and they calculate the pagerank values of total graph after certain time period, which takes long execution time and can be in days. In the same way updating the graph also compels to recalculate all the pagerank values of all the pages in the graph. This suggests possible applicability of the distributed algorithm to pagerank computations as a replacement for the centralized pagerank calculation algorithm. Considering the importance of the "Ranking" in searching context, our approach DYNA-RANK, focuses upon efficiently calculating and updating Google's pagerank vector using "peer to peer" system. The changes in the web structure will be handled incrementally amongst the peers. DYNA-RANK produces the relative pagerank on each peer. DYNA-RANK is proven to take less computation time and less number of iterations compared to centralized approach.

Keywords: *DYNA-RANK, PageRank, Peer to peer system, Power method, BlockRank, Web graph, ObjectRank, HubRank.*

1. Introduction

The increasing size and the changing structure of the Web or alike huge social networks makes the task of searching more complex and crucial. Searching in these types of structures will give a large set of results for a single query, which can be up to millions in number containing both relevant and irrelevant results. So the ranking plays important role for making Search Engines relevant. As the browsing of the database or graph is primly dependent on the ranking of the results of the search query, there are many applications of a good ranking technique such as providing good search results and predicting the web traffic. So the issues related to the ranking of the entities are very important. Fast and up to date computation of pageranks [8] has become mandatory for good ranking. In available methods overhead of the calculation of pageranks and also updation of pageranks after modification of web graph (modification may be insertion or deletion of both link and page) is very high.

Figure 1 shows propagation of pageranks. The value of the pagerank decreases with each propagation and at one point of iteration it becomes negligible. This negligible factor is the factor of convergence, we assume it as $epsilon$. The pageranks can be calculated by the Power Method [6, 1] as follows,

$$repeat\ until\ P_r\ converges$$

$$P_r = \alpha \cdot C \cdot P_r + (1-\alpha) \cdot r \qquad \cdots [1]$$

where P_r is pagerank vector, $(1-\alpha)$ is teleport probability such that $0 < \alpha < 1$, r is a teleport vector which must be nonnegative and $|r|_1 = 1$. C is conductance matrix of the graph G. The entry $C(j,i)$ is defined as the probability of walking to node j from node i following the edges $i \rightarrow j$. So the columns of C must add up to 1. Higher the $C(j,i)$ value more is the probability of walking to node j from node i and low resistance to authority flow from i to j.

The available technique for calculating distributed pagerank is given in [9]. This technique distributes the graph nodes on peers in random fashion which increases the network traffic. It also uses crawling for calculation of pageranks instead of Power Method. So it takes more time for calculating pageranks and increases network traffic.

This motivates to minimize the computation time and updation time of the pagerank. This work will provide a method for calculating and updating pageranks, which

978-0-7695-3308-7/08 $25.00 © 2008 IEEE
DOI 10.1109/ICCSIT.2008.118

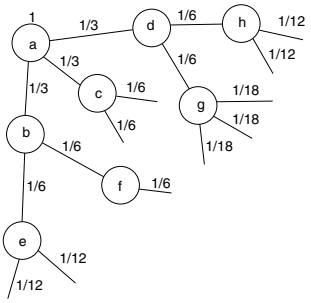

Figure 1. Propagation of pagerank

we call collectively as DYNA-RANK. We have proposed a algorithm for computation named DYNA-RANK, using peer to peer system. The implementation is based on asynchronous (chaotic) iterative solution. The "peer to peer" implementation also enables incremental updating of pageranks, as new documents are entered into or deleted from the network. Incremental update gives continuously accurate pageranks, whereas the currently centralized web crawl and computation of pageranks of internet documents requires several days.

2. Related Work

The ranking method, PageRank [8] is playing a conceptual and important role even for the search engine giants like Google, which uses PageRank as the major parameter for the ranking along with some strong information retrieving techniques. All of the methods for ranking like Personalized PageRank [4], ObjectRank [2], HubRank [3] are based on PageRank concept. Even though having all these methods and many more, efficient updation of the PageRank is very important. In Blockrank [5], it has been shown that, the hyperlink graph of the web has a nested block structure. In Blockrank [5], for all the hyperlinks in FULLLARGE WEB, it counts how many of these links are "intra-host" links (links from a page to another page in the same host) and how many are "interhost" links (links from a page to a page in a different host). It shows that 79.1% of the links in this dataset are intra-host links, and 20.9% are inter-host links. It also investigates the number of links that are intra-domain links, and the number of links that are inter-domain links and shows that an even larger majority of links are intradomain links (83.9%).

3. Problem Description

The problem can be stated as to minimize the computation time for calculation and updation of pagerank vector for large web structure.

The web structure forms a conductance matrix. Given a large size web structure G, which forms the conductance matrix C of size $n \times n$, where n is the number of pages in the web as input along with the teleport probability $1 - \alpha$ and initializing teleport vector r to $1/n$, that is the equal probabilities to all nodes and the P_r to the value of equal weights to all the nodes that is 1.

The aim of the work is to output the pagerank values in minimum computation time as well as handling the updates in the web graph and updating the pageranks of new graph in minimum computation time.

Summarizing the problem as to compute the pagerank of web graph by power method that is $P_r = \alpha \cdot C \cdot P_r + (1-\alpha) \cdot r$ using minimum time by distributing the graph on peers into sub-graphs and calculating the pagerank on each peer and then propagating only the update messages to the connected peers. Pagerank vector will be the eigen vector for the major eigen value (i.e. largest eigen value) for matrix C.

4. DYNA-RANK

In DYNA-RANK, we extend the same concept of decrease in the effect of the pageranks with propagation, to minimize the computations for updating. If some of the part of web gets updated then pageranks of whole graph may not change, which means the changes will affect up to certain domain. For example, changes in the host "www.indianrail.gov.in" need not have any effect in pagerank of "www.stanford.com", which clearly makes the pagerank calculation of the stanford host unnecessary.

Web graph of the dataset is distributed amongst the peers in DYNA-RANK. The investigation of BlockRank [5] can be used for distribution of the web graph on peers, such as distributing each domain or few hosts on one peer. This sub-graph we call as peer-graph. The inlinks and outlinks of the structure are maintained for each peer. For example, the structure of distribution can be visualized in Figure 2, where the domain X is on peer 1, domain Y on peer 2, and domain Z on peer 3. The intra-peer and inter-peer links are shown in bold and dotted lines respectively. Algorithm 1 explains all the steps of DYNA-RANK.

Algorithm 1: DYNA-RANK
/* Distribute the total Web Graph on to each Peer based on the Domain names or hosts, this is one time job which will last long. We keep track of all the inlinks and outlinks from each peer. Assuming that the graph is distributed and the inlinks and outlinks information is supplied before applying this algorithm */

$Input$: Conductance matrices of web peer-graphs distributed on peers
$Output$: The pageranks of web graph nodes on each peer

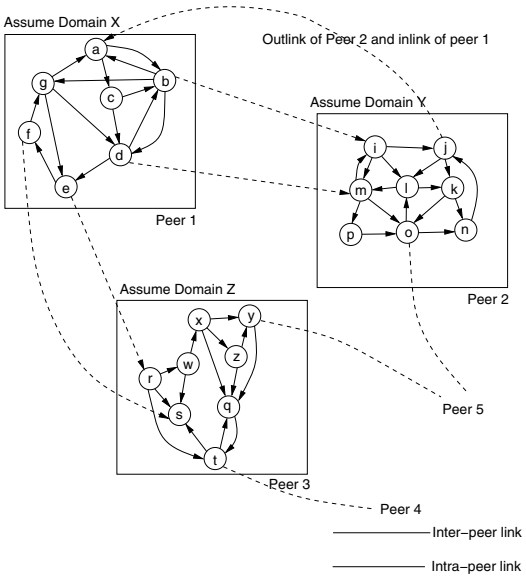

Figure 2. Distribution of the web graph

$Notations$:
$Inlink$: Link coming to the peer-graph from other peer
$Outlink$: Link going out of the peer-graph to other peer
no_of_peers: Total number of peers
n_i: number of nodes in peer-graph in $Peer(i)$
$epsilon$: a minimum threshold
Steps:

1. **BEGIN**
2. **Initialization:** [concurrently on each peer]
   a. Assign numbers or names to the peers (assume here we have assigned integer numbers from 0 to no_of_peers)
   b. Initialize all the DYNA-RANK vectors in all the peers (for all peer-graph) to 1
   c. Initialize all the teleport vectors in all the peers (for all peer-graph) to $1/n_i$
   d. The value of the teleport probability is 0.15 [8] (i.e. α=0.85)
   e. Ignore the inlinks and outlinks in the peer-graph (assume the weights of both as 0 at initialization)
3. **Calculation of pagerank:** [concurrently on each peer]
   a. Calculate pagerank of individual peer-graph by Power Method [6, 1].
4. **Incremental updating and propagating pageranks:**
   a. Starting from $peer(i)$, where i is any node to all the connected peers by outlinks until there are no more update messages on network repeat through e.
   b. Check for all inlinks and if the weight of inlink is modified then add that weight to the pagerank value of the destination nodes on $peer(i)$ (that is this peer)
   c. Calculate the pagerank of the peer-graph by power method and update the pagerank vector.
   d. Calculate the outlink weights for all the outlinks by

$$new\ weight(K, L) = \frac{P_R(K)}{(n(K)_{peer(i)}) + 1}$$

   Where,
   $node(K)$ is on $peer(i)$ and $node(L)$ is on $peer(j)$ and $(i \neq j)$, $P_R(K)$ is pagerank of $node(K)$, $n(K)_{peer(i)}$ is number of outlinks of $node(K)$ on $peer(i)$, $new\ weight(i, j)$ is New outlink weight for edge $node(k) \rightarrow node(L)$.
   e. for all the outlinks having $relative\ change(RC) \geq epsilon$ send the new outlink weights to the peers connected to $peer(i)$ by outlinks . Where,

$$RC = \frac{abs(new\ weight - old\ weight)}{(new\ weight)}$$

5. **Propagation of incremental updates:**
   a. Whenever a peer-graph gets changed (changes may be link modifications or the node addition or the node deletion) then assume that peer as $peer(i)$ and continue with the step 4
6. **END**

Step 2 of DYNA-RANK algorithm initializes the outlink and inlink weights as well as the initial pagerank vector for all peers. While initialization we ignore inlinks and outlinks weights, which is assumed to be 0 weights. The pagerank vectors will be initialized to 1 and teleport vector to $1/n$, where n number of nodes on the peer. This means, at the initialization we are assuming that all the nodes are having equal probability of starting the random surfing. In Figure 2 the peers 1, 2, 3 will be initialized.

Step 3 will calculate the pageranks of individual peer-graphs of domain X, Y, Z. The computation is done by power method given in Equation (1) concurrently on all peers, which will modify the current pagerank values.

Step 4 is the propagation of pagerank updates to the neighboring peers. In this step we start from any of the peer i, this peer will check inlink weights for any updations, then it will add the values of the inlink weights to the pageranks

of the destination nodes on peer i. Then calculate the pagerank vector for this peer with power method and update outlink weights for peer i. If the outlink weights are increased or decreased by more than or equal to threshold ($epsilon$) we send new outlink weight to all connected peers. In the example of Figure 2, we start from peer 1. The inlinks are checked for updates and the weights on inlinks are added to respective nodes on peer 1. However peer 1 will calculate the pageranks on its peer-graph and update the outlinks. This peer will send the updates to the peer 2,3. The same procedure will be carried on both peers 2,3.

The $epsilon$ is the factor of great importance as the network traffic and accuracy are dependent on it. If we decrease its value the network traffic will increase and accuracy will be high. Increasing $epsilon$ will decrease the accuracy and network traffic both. So we need to have practical values of time related to different epsilon to balance the network traffic and accuracy.

First three steps of the algorithm are performed only once. Then the fourth step will finalize the pageranks in its first pass. Up to this point, graph is considered to have no updates. Once the static pageranks of all the peer-graphs are finalized, then the most important part is calculation of pageranks dynamically when the graph gets modified.

Step 5 of the DYNA-RANK algorithm is a important step which handles the modifications. Assuming the modifications will be reflected in the peer-graph automatically, as soon as the graph changes (may be link or node change) the "Incremental updating and propagating pageranks" step will be repeated. Computation of the pagerank will be invoked for all updates and the procedure will continue from step 3. Different documents will attain their final pagerank at different times. When any of the peer-graphs X, Y, Z changes, the pageranks will be recalculated and the fourth step will be continued.

5. Experimental Study

All the experiments are performed on Windows systems 2.8 GHz Pentium PCs using JDK 6. This setup is peer to peer system having three PCs connected by 100 Mbps LAN.

The inputs are the conductance matrices stored in text files. Peer to peer outlinks as source node, destination node are stored in text file on each peer. Description of the dataset on each peer is given in Table 1. In the dataset other links are ignored for the experimental purpose and only inlinks and outlinks of the three peer-graphs amongst themselves are considered.

The experiments are carried out considering different values of the threshold ($epsilon$) which is taken as 0.01, 0.001, 0.0001, 0.00001. The decreasing epsilon value will give more accurate pageranks and takes more time as well as iterations for computation.

Table 1. Dataset used

Peer	Number of nodes	Inlinks	Outlinks	Intralinks
1	105	68	82	444
2	103	72	55	296
3	119	65	68	343

The performance of the DYNA-RANK is compared with centralized approach using pagerank calculation and results for the given dataset specified above, are quoted in following tables.

Table 2. Computation time

Threshold ($epsilon$)	Centralized Approach (time in milli Sec)	DYNA-RANK Method (time in milli Sec)		
		Peer1	Peer2	Peer3
0.01	9641	4000	3781	3297
0.001	11250	4563	3234	2938
0.0001	12906	5078	4500	3343
0.00001	14578	5656	4437	4515

Table 3. Number of iterations

Threshold ($epsilon$)	Centralized Approach (no. of iterations)	DYNA-RANK Method (no. of iterations)		
		Peer1	Peer2	Peer3
0.01	54	47	47	48
0.001	68	61	61	62
0.0001	82	75	75	76
0.00001	96	89	89	90

Table 2 and 3 shows that the DYNA-RANK performs better compared to centralized method in computation time as well as iterations. DYNA-RANK is better, as the updation of all the pageranks are incremental and dynamic, no updates are ignored or kept pending for long and all the fresh pagerank values will be available for use. The network traffic of the distributed pagerank [9] is approximately of logarithmic order to the threshold. Network traffic in DYNA-RANK will be less than the logarithmic order as the distribution of the graph is done on the basis of the observation stated in [5], which shows that approximately 80% of the links are intradomain (or intrahost) and it can be observed in Table 1, so the communication in between peers will be minimum. DYNA-RANK is better in time complexity because, we compute the pagerank for all the nodes in incremental manner which gives the results faster. It is only calculated and propagated to next peers till the update value is above some threshold ($epsilon$), which can be balanced to give better performance in network and accu-

racy. This makes it to perform better than some available techniques [9, 7]. The performance graphs of the DYNA-RANK with the centralized approach are given in Figure 3 and Figure 4.

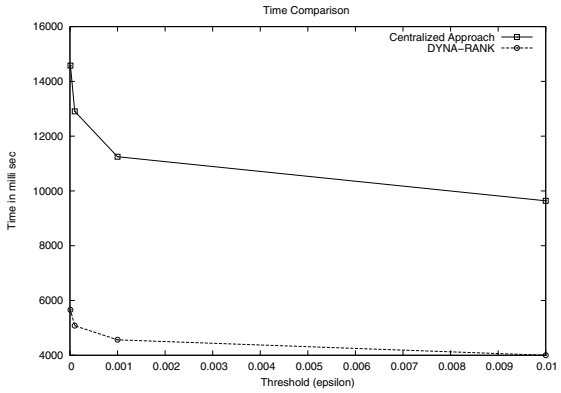

Figure 3. Time comparison graph

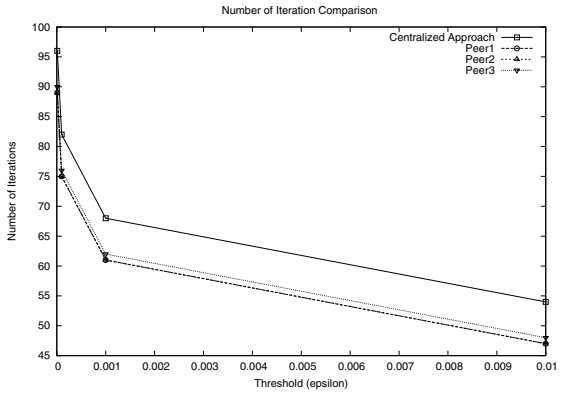

Figure 4. Iteration comparison graph

Graph in Figure 3 shows the time comparison of the centralized approach for pagerank computation by Power Method versus DYNA-RANK method. Here, the time for DYNA-RANK is taken as the maximum time taken by any of the peers. We can see that as the $epsilon$ value increases, time for computation is less and the number of messages as well as the iterations are less. The graph in Figure 4 shows the iteration comparison of the three peers with the centralized Power Method for PageRank computation, which clearly indicates that increase in the $epsilon$ value decreases iteration count.

6. Conclusion

The DYNA-RANK is the initiation of the pagerank calculation in new directions with new ideas incorporated. It is shown that, it practically minimizes the time required for calculating and updating the pagerank vector. In the experimental results for a graph of size 327 nodes the time is minimized by approximately 58% , 59%, 60%, 61% and the number of iterations are reduced by approximately 11%, 8%, 7%, 6%, when threshold was 0.01, 0.001, 0.0001, 0.00001 respectively with minimum network traffic. In DYNA-RANK, sum of all pagerank values will be equal to the number of peers used. Each of the peer will be running Power Method which gives summation of pageranks as 1 to each peer and still preserves the relative probability of pages.

7. Future work

Here, we have only considered general pagerank, where all the outlinks are having equal probability of being accessed. The proposed work can be further extended for Personalized pagerank. This work can also be extended to include the load balancing of web graph and distribution can be done taking the affinity of the hosts and domains into consideration.

References

[1] E. Andersson and P.-A. Ekstrom. Investigating google's pagerank algorithm. *A Technical Report in Scientific Computing*, 2004.

[2] A. Balmin, V. Hristidis, and Y. Papakonstantinou. Objectrank:authority-based keyword search in databases. *Proceedings of the Thirtieth International Conference on Very Large Data Bases*, pages 564–575, August 2004.

[3] S. Chakrabarti. Dynamic personalized pagerank in entity-relation graphs. *WWW 2007: Proceedings of the 16th international conference on World Wide Web*, pages 571–580, January 2007.

[4] G. Jeh and J. Widom. Scaling personalized web search. *In WWW Conference*, pages 271–279, January 2003.

[5] S. D. Kamvar, T. H. Haveliwala, C. D. Manning, and G. H. Golub. Exploiting the block structure of the web for computing. 2003.

[6] A. N. Langville and C. D. Meyer. Deeper inside pagerank. *Internet Mathematics*, (1(3)):335–380, January 2004.

[7] A. N. Langville and C. D. Meyer. Updating pagerank with iterative aggregation. *WWW2004*, May 2004.

[8] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. *Stanford Digital Libraries Working Paper*, January 1998.

[9] K. Sankaralingam, S. Sethumadhavan, and J. C. Browne. Distributed pagerank for p2p systems. *Appears in the* 12^{th} *International Symposium on High Performance Distributed Computing*, 2003.