A REVIEW ON: DYNAMIC LINK BASED RANKING

D Nagamalleswary , A. Ramana Lakshmi
Department of Computer Science & Engineering, PVP Siddhartha Institute of Technology
Vijayawada, Andhra Pradesh, India

ABSTRACT

Dynamic authority-based ranking methods such as personalized PageRank and ObjectRank. Since they dynamically rank nodes in a data graph using an expensive matrix-multiplication method, the online execution time rapidly increases as the size of data graph grows. ObjectRank spends 20-40 seconds to compute query-specific relevance scores, which is unacceptable. We introduce a novel approach, BinRank, that approximates dynamic link-based ranking scores efficiently. BinRank partitions a dictionary into bins of relevant keywords and then constructs materialized subgraphs (MSGs) per bin in preprocessing stage. In query time, to produce highly accurate top-K results efficiently, BinRank uses the MSG corresponding to the given keyword, instead of the original data graph. In this project, a BinRank system that employs a hybrid approach where query time can be traded off for preprocessing time and storage. BinRank closely approximates ObjectRank scores by running the same ObjectRank algorithm on a small subgraph, instead of the full data graph.

***KEYWORDS*:** *Online keyword search, Object Rank, scalability, approximation algorithms*

I. INTRODUCTION

The PageRank algorithm[1] utilizes the Web graph link structure to assign global importance to Web pages. It works by modeling the behavior of a "random Web surfer" who starts at a random Web page and follows outgoing links with uniform probability. The PageRank score is independent of a keyword query. Recently, dynamic versions of the PageRank algorithm have become popular. They are characterized by a query-specific choice of the random walk starting points.

In particular, two algorithms have got a lot of attention: Personalized PageRank (PPR) for Web graph data sets[2] [3] [4] [5] and ObjectRank for graph-modeled databases[6] [7] [8] [9] [10]. PPR is a modification of PageRank that performs search personalized on a preference set that contains Web pages that a user likes. For a given preference set, PPR performs a very expensive fixpoint iterative computation over the entire Web graph, while it generates personalized search results.[3] [4] [5] Therefore, the issue of scalability of PPR has attracted a lot of attention. ObjectRank [6]extends (personalized) PageRank to perform keyword search in databases. ObjectRank uses a query term posting list as a set of random walk starting points and conducts the walk on the instance graph of the database. The resulting system is well suited for "high recall" search, which exploits different semantic connection paths between objects in highly heterogeneous data sets.

ObjectRank has successfully been applied to databases that have social networking components, such as bibliographic data and collaborative product design. However, ObjectRank suffers from the same scalability issues as personalized PageRank, as it requires multiple iterations over all nodes and links of the entire database graph. The original ObjectRank system has two modes: online and offline. The online mode runs the ranking algorithm once the query is received, which takes too long on large graphs. For example, on a graph of articles of English Wikipedia1 with 3.2 million nodes and 109 million links, even a fully optimized in-memory implementation of ObjectRank takes 20-50 seconds to run. In the offline mode, ObjectRank precomputes top-k results for a query workload in advance.

This precomputation is very expensive and requires a lot of storage space for precomputed results. Moreover, this approach is not feasible for all terms outside the query workload that a user may search for, i.e., for all terms in the data set dictionary. For example, on the same Wikipedia data set, the full dictionary precomputation would take about a CPU-year.

II. EXISTING SYSTEM

- PageRank algorithm utilizes the Web graph link structure to assign global importance to Web pages It works by modeling the behavior of a "random Web surfer" who starts at a random Web page and follows outgoing links with uniform probability.
- The PageRank score is independent of a keyword query.
- Personalized PageRank (PPR) for Web graph data sets and ObjectRank for graph-modeled databases results. Therefore, the issue of scalability of PPR has attracted a lot of attention.
- ObjectRank extends (personalized) PageRank to perform keyword search in databases. ObjectRank uses a query term posting list as a set of random walk starting points and conducts the walk on the instance graph of the database.

III. PROPOSED SYSTEM

- In this project, a BinRank system that employs a hybrid approach where query time can be traded off for preprocessing time and storage. BinRank closely approximates ObjectRank scores by running the same ObjectRank algorithm on a small subgraph, instead of the full data graph.
- BinRank query execution easily scales to large clusters by distributing the subgraphs between the nodes of the cluster.
- We are proposing the BinRank algorithm for the trade time of search. Our alogorithm solves the time consuming problem in query execution. Time will be reduced because of cache storage and redundant query handling method.

IV. BIN CONSTRUCTION

As outlined above, we construct a set of MSGs for terms of a dictionary or a workload by partitioning the terms into a set of term bins based on their co-occurrence. We generate an MSG for every bin based on the intuition that a sub graph that contains all objects and links relevant to a set of related terms should have all the information needed to rank objects with respect to one of these terms.

There are two main goals in constructing term bins. First, controlling the size of each bin to ensure that the resulting sub graph is small enough for Object Rank to execute in a reasonable amount of time. Second, minimizing the number of bins to save the preprocessing time. After all, we know that pre computing Object Rank for all terms in our corpus is not feasible. To achieve the first goal, we introduce a max Bin Size parameter that limits the size of the union of the posting lists of the terms in the bin, called bin size. As discussed above, Object Rank uses the convergence threshold that is inversely proportional to the size of the base set, i.e., the bin size in case of sub graph construction. Thus, there is a strong correlation between the bin size and the size of the materialized sub graph. The value of max Bin Size should be determined by quality and performance requirements of the system. The problem of minimizing the number of bins is NPhard. In fact, if all posting lists are disjoint, this problem reduces to a classical NP-hard bin packing problem [12]. We apply a greedy algorithm that picks an unassigned term with the largest posting list to start a bin and loops to add the term with the largest overlap with documents already in the bin. We use a number of heuristics to minimize the required number of set intersections, which dominate the complexity of the algorithm. The tight upper bound on the number of set intersections that our algorithm needs to perform is the number of pairs of terms that co-occur in at least one document. To speed-up the execution of set intersections for larger posting lists, we use KMV synopses [13] to estimate the size of set intersections.

The algorithm in Fig. 1 works on term posting lists from a text index. As the algorithm fills up a bin, it maintains a list of document IDs that are already in the bin, and a list of candidate terms that are known to overlap with the bin (i.e., their posting lists contain at least one document that was already

placed into the bin). The main idea of this greedy algorithm is to pick a candidate term with a posting list that overlaps the most with documents already in the bin, without posting list union size exceeding the maximum bin size.

V. ALGORITHM

Input: A set of workload terms P with their position lists
Output: A set of bins B

```
(1) while P is not empty do
        create a new empty bin b then create a empty cache of candidate terms C
(2) pick term t € P with the largest positioning list size |t|
(3) while t is not null do
    add t to b and remove it from W then compute a set of terms T that co-occurance with t
(4) for each t' € T do
    insert mapping <t', null> into C
    end for each
(5) best I:=0
    for each mapping <c,i> € C do
    if i= null then // b ∩ c has not been computed yet
    i := |b∩c| then update mapping <c,i> from C
    end if
(6) union:=|b|+|c|-i
    if uninon union >Max Bin Size then remove <c, i> from C
    else if i> best of I then best I:=I , t :=c
    end if
    end for each
(7) if best I:=0 then
    pick t € P with maximium |t| ≤ maxBinsize -|b|
    if no such t exists , t:=null
    end if
    end while
    add completed to B
        end while
```

fig1: bin algorithm

While it is more efficient to prepare bins for a particular workload that may come from a system query log, it is dangerous to assume that a query term that has not been seen before will not be seen in the future. We demonstrate that it is feasible to use the entire data set dictionary as the workload, in order to be able to answer any query.

Due to caching of candidate intersection results in lines 12- 14 of the algorithm, the upper bound on the number of set intersections performed by this algorithm is the number of pairs of co-occurring terms in the data set. Indeed, in the worst case, for every term t that has just been placed into the bin, we need to intersect the bin with every term to that co occurs with t, in order to check if t0 is subsumed by the bin completely, and can be placed into the bin "for free."

For example, consider N terms with posting lists of size X each that all co-occur in one document d0 with no other co-occurrences. If maximum bin size is 2(X - 1), a bin will have to be created for every term. However, to get to that situation, our algorithm will have to check intersections for every pair of terms. Thus, the upper bound on the number of intersections is tight.

In fact, it is easy to see from the above example that no algorithm that packs the bins based on the maximum overlap can do so with fewer than N(N – 1)/2 set intersections in the worst case. Fortunately, real-world text databases have structures that are far from the worst case.

VI. SYSTEM ARCHITECTURE

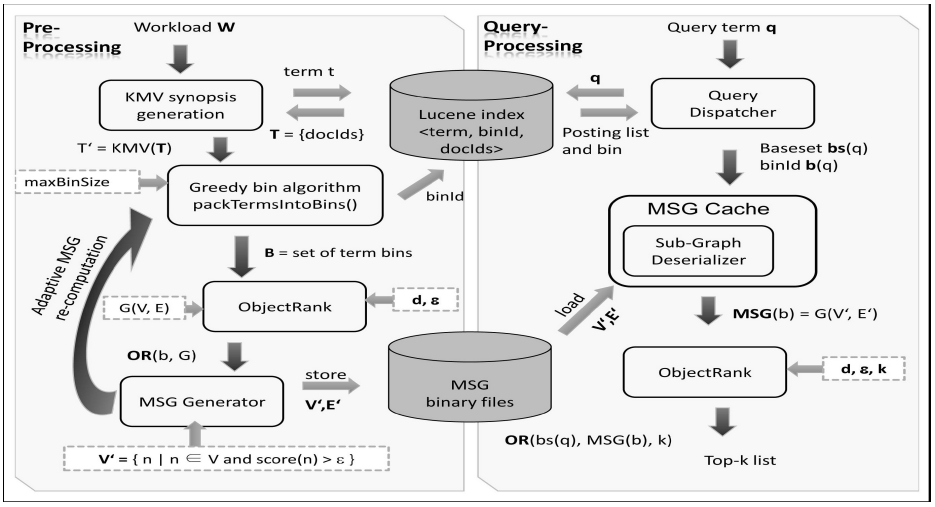

Fig2: Binranking System Architecture

Fig. 2 shows the architecture of the Bin Rank system. During the preprocessing stage (left side of figure), we generate MSGs. During query processing stage (right side of figure), we execute the Object Rank algorithm on the sub graphs instead of the full graph and produce high-quality approximations of top-k lists at a small fraction of the cost. In order to save preprocessing cost and storage, each MSG is designed to answer multiple term queries. We observed in the Wikipedia data set that a single MSG can be used for 330-2,000 terms, on average.

6.1 Preprocessing

The preprocessing stage of Bin Rank starts with a set of workload terms W for which MSGs will be materialized. If an actual query workload is not available, W includes the entire set of terms found in the corpus. We exclude from W all terms with posting lists longer than a system parameter max Posting List. The posting lists of these terms are deemed too large to be packed into bins. We execute Object Rank for each such term individually and store the resulting top-k lists. Naturally, max Posting List should be tuned so that there are relatively few of these frequent terms. In the case of Wikipedia, we used max Posting List=2,000 and only 381 terms out of about 700,000 had to be pre computed individually. This process took 4.6 hours on a single CPU.

For each term w € W, Bin Rank reads a posting list T from the Lucene3 index and creates a KMV synopsis T0 that is used to estimate set intersections. The bin construction algorithm, Pack Terms Into Bins, partitions W into a set of bins composed of frequently co-occurring terms. The algorithm takes a single parameter max Bin Size, which limits the size of a bin posting list, i.e., the union of posting lists of all terms in the bin. During the bin construction, Bin Rank stores the bin identifier of each term into the Lucene index as an additional field. This allows us to map each term to the corresponding bin and MSG at query time.

The Object Rank module takes as input a set of bin posting lists B and the entire graph G(V,E) with a set of Object Rank parameters, the damping factor d, and the threshold value €. The threshold determines the convergence of the algorithm as well as the minimum Object Rank score of MSG nodes.

Our Object Rank implementation stores a graph as a row compressed adjacency matrix. In this format, the entire Wikipedia graph consumes 880 MB of storage and can be loaded into main memory for MSG generation. In case that the entire data graph does not fit in main memory, we can apply parallel Page Rank computation techniques such as hyper graph partitioning schemes described in [14].

6.1.1 Steps

I. User Registration
II. Authentication Module
III. Search - Query Submission

IV. Index Creation
V. BinRank Algorithm Implementation Graph based on Rank

6.1.2 Data Flow Diagram

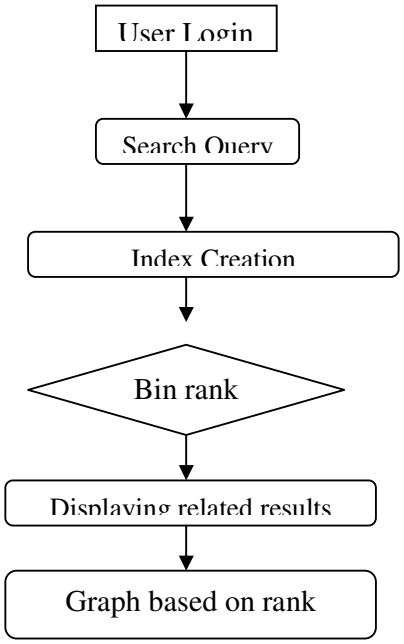

Fig3: Data Flow Processing Of Binranking

6.2 Query Processing

For a given keyword query q, the query dispatcher retrieves from the Lucene index the posting list *bs (q)* (used as the base set for the Object Rank execution) and the bin identifier *b(q).* Given a bin identifier, the MSG map per determines whether the corresponding MSG is already in memory. If it is not, the MSG de-serialize reads the MSG representation from disk. The Bin Rank query processing module uses all available memory as an LRU cache of MSGs.

For smaller data graphs, it is possible to dramatically reduce MSG storage requirements by storing only a set of MSG nodes V 0, and generating the corresponding set of edges E0 only at query time. However, in our Wikipedia, data set that would introduce an additional delay of 1.5-2 seconds, which is not acceptable in a keyword search system.

The Object Rank module gets the in-memory instance of MSG, the base set, and a set of Object Rank calibrating parameters: 1) the damping factor d; 2) the convergence threshold ė; and 3) the number of top-k list entries k. Once the Object Rank scores are computed and sorted, the resulting document ids are used to retrieve and present the top-k objects to the user.

VII. CONCLUSION

In this paper, we proposed BinRank as a practical solution for scalable dynamic authority-based ranking. It is based on partitioning and approximation using a number of materialized subgraphs. We showed that our tunable system offers a nice trade-off between query time and preprocessing cost.

We introduce a greedy algorithm that groups co-occurring terms into a number of bins for which we compute materialized subgraphs. Note that the number of bins is much less than the number of terms. The materialized subgraphs are computed offline by using ObjectRank itself. The intuition behind the approach is that a subgraph that contains all objects and links relevant to a set of related terms should have all the information needed to rank objects with respect to one of these terms. Our extensive experimental evaluation confirms this intuition. For future work, we want to study the impact of other keyword relevance measures, besides term co-occurrence, such as thesaurus or ontologies, on the performance of BinRank. By increasing the relevance of keywords in a bin, we expect the quality of

materialized subgraphs, thus the top-k quality and the query time can be improved. We also want to study better solutions for queries whose random surfer starting points are provided by Boolean conditions. And ultimately, although our system is tunable, the configuration of our system ranging from number of bins, size of bins, and tuning of the ObjectRank algorithm itself (edge weights and thresholds) is quite challenging, and a wizard to aid users is desirable.

VIII. FUTURE WORK

To further improve the performance of BinRank, we plan to integrate BinRank and HubRank [8] by executing HubRank on MSGs BinRank generates. Currently, we use the ObjectRank algorithm on MSGs in query time. Even though HubRank is not as scalable as BinRank, it performs better than ObjectRank on smaller graphs such as MSGs. In this way, we can leverage the synergy between BinRank and HubRank

REFERENCES

[1] J. Cho and U. Schonfeld, "Rankmass Crawler: A Crawler with High PageRank Coverage Guarantee," Proc. Int'l Conf. Very Large Data Bases

[2]R. Fagin, R. Kumar, M. Mahdian, D. Sivakumar, and E. Vee, "Comparing and aggregating rankings with ties," in PODS '04.LDB), 2007.

[3] H. Hwang, A. Balmin, B. Reinwald, and E. Nijkamp, "Binrank: Scaling dynamic authority-based search using materialized subgraphs," in ICDE '09, 2009, pp. 66–77.

[4] G. Jeh and J. Widom, "Scaling personalized web search," in WWW '03. New York, NY, USA: ACM, 2003, pp. 271–279

[5]. Ding, L., Pan, R., Finin, T.W., Joshi, A., Peng, Y., Kolari, P.: Finding and ranking knowledge on the semantic web. In: Proceedings of the International Semantic Web Conference. (2005) 156{170

[6] Hogan, A., Harth, A., Decker, S.: Reconrank: A scalable ranking method for semantic web data with context. In: Proceedings of Second International Workshop on Scalable Semantic Web Knowledge Base Systems, Athens, GA, USA. (11 2006)

[7] S. Brin and L. Page, "The Anatomy of a Large-Scale Hypertextual Web Search Engine," Computer Networks, vol. 30, nos. 1-7, pp. 107- 117, 1998.

[8] T.H. Haveliwala, "Topic-Sensitive PageRank," Proc. Int'l World Wide Web Conf. (WWW), 2002.

[9] G. Jeh and J. Widom, "Scaling Personalized Web Search," Proc. Int'l World Wide Web Conf. (WWW), 2003.

[10] D. Fogaras, B. Ra´cz, K. Csaloga´ny, and T. Sarlo´ s, "Towards Scaling Fully Personalized PageRank: Algorithms, Lower Bounds, and Experiments," Internet Math., vol. 2, no. 3, pp. 333-358, 2005.

[11] K. Avrachenkov, N. Litvak, D. Nemirovsky, and N. Osipova, "Monte Carlo Methods in PageRank Computation: When One Iteration Is Sufficient," SIAM J. Numerical Analysis, vol. 45, no. 2, pp. 890-904, 2007.

[12] A. Balmin, V. Hristidis, and Y. Papakonstantinou, "ObjectRank: Authority-Based Keyword Search in Databases," Proc. Int'l Conf. Very Large Data Bases (VLDB), 2004.

[13] Z. Nie, Y. Zhang, J.-R. Wen, and W.-Y. Ma, "Object-Level Ranking: Bringing Order to Web Objects," Proc. Int'l World Wide Web Conf. (WWW), pp. 567-574, 2005.

[14] S. Chakrabarti, "Dynamic Personalized PageRank in Entity- Relation Graphs," Proc. Int'l World Wide Web Conf. (WWW), 2007.

[15] H. Hwang, A. Balmin, H. Pirahesh, and B. Reinwald, "Information Discovery in Loosely Integrated Data," Proc. ACM SIGMOD, 2007.

[16] V. Hristidis, H. Hwang, and Y. Papakonstantinou, "Authority- Based Keyword Search in Databases," ACM Trans. Database Systems, vol. 33, no. 1, pp. 1-40, 2008.

[17] M. Kendall, Rank Correlation Methods. Hafner Publishing Co., 1955.[12] M.R. Garey and D.S. Johnson, "A 71/60 Theorem for Bin Packing," J. Complexity, vol. 1, pp. 65-106, 19

[18] M.R. Garey and D.S. Johnson, "A 71/60 Theorem for Bin Packing," J. Complexity, vol. 1, pp. 65-106, 1985.

[19] K.S. Beyer, P.J. Haas, B. Reinwald, Y. Sismanis, and R. Gemulla,"On Synopses for Distinct-Value Estimation under Multiset Operations," Proc. ACM SIGMOD, pp. 199-210, 2007.

[20] J.T. Bradley, D.V. de Jager, W.J. Knottenbelt, and A. Trifunovic, "Hypergraph Partitioning for Faster Parallel PageRank Computation,"Proc. Second European Performance Evaluation Workshop (EPEW), pp. 155-171, 2005.

[21] , L., Brin, S., Motwani, R., Winograd, T.: The pagerank citation ranking: Bringing order to the web. Technical Report 1999-66, Stanford InfoLab (1999)

Authors Biographies

D. Nagamalleswary pursuing M.Tech from P.V.P.S.I.T and received B.Tech from Nimra engineering college. She currently is working as Assist. Professor in K.L University.

A. Ramna Lakshmi pursuing P.hd and she is currently working as Associate Professor in PVP Siddhartha Institute of Engineering and Technology, Kanuru.