Search Result Clustering via Randomized Partitioning of Query-Induced Subgraphs

Aleksandar Bradic
Faculty of Electrical Engineering, Belgrade
Email: abradic@acm.org

Abstract—In this paper, we present an approach to search result clustering, using partitioning of underlying link graph. We define the notion of "query-induced subgraph" and formulate the problem of search result clustering as a problem of efficient partitioning of given subgraph into topic-related clusters. Also, we propose a novel algorithm for approximative partitioning of such graph, which results in cluster quality comparable to the one obtained by deterministic algorithms, while operating in more efficient computation time, suitable for practical implementations. Finally, we present a practical clustering search engine developed as a part of this research and use it to get results about real-world performance of proposed concepts.

Index Terms—Information Search and Retrieval, Graph Clustering, Randomized Algorithms, Web Measurement

I. INTRODUCTION

Efficient representation of search results poses a significant challenge for modern search engines. The widely-accepted score-based model [1], although quite effective in the general case of search for the best document matching the given query, is usually insufficient in situations which require representation of larger set of relevant results. This is especially true in the case of clustering and exploratory search engines, which focus not only on representation of the relevance, but also the way the results are related and their organization into clusters of related documents.

Clustering based on document information content has been a well studied topic in Information Retrieval (IR). Standard IR clustering methods, based on the cluster hypothesis [2], usually operate by calculating appropriate content-based relevance values and imposing certain similarity metric, have been accepted by Search Engine community and implemented in a number of real-world clustering engines (Vivisimo, Carrot Clustering Engine, Mooter, Clusty). Still, we can observe that clustering Web data in this manner fails to capture the essential component of Web documents, which is the hyperlink information, reflected in link graph, which describes the explicit way in which the documents are connected. A lot of algorithms utilize this structure to extract information about document relevance (PageRank [1]), and community structure (HITS [3]). Great success of these algorithms, indicated the significance of link structure in Web data analysis, and suggested extension of such concept to other related problems, like the problem of community detection [3], and Web data clustering [4]. However, although there are significant results in the area of link-based Web data clusterings, implementing such algorithms in practical search engine still poses a significant challenge, primarily due to the fact that, unlike IR-based methods, which operate on set of values precomputed for each document, graph-based algorithms operate on dynamical query-dependent representation of entire link graph, which makes precomputation impossible and problem both computationally and space-intensive. As a result of this, currently there are no real-world clustering engines that implement search result clustering using the link-graph approach.

In this paper, we propose a relaxation of the problem of search result clustering from the problem of clustering the entire graph to the domain of query-induced subgraph, representing a subgraph generated by given search query and show the validity of such proposal by determining that the essential structural properties of the entire graph are still preserved in given subgraph. Further, we propose a novel algorithm for approximative clustering of such subgraphs, which enables us more space and computationally efficient clustering, with variable margin of error, suitable for implementation in real-world search engines. Finally, we present a search engine called randomNode, implemented as a part of this research, which demonstrates usability of proposed concepts in real-world application.

II. RELATED WORK

In [5], authors perform the first analysis of the general structure of the Web, and determine that node degree distribution follows a simple power-law of the form \( k^{-\theta} \), with \( \theta = 2.1 \) for in-degree and \( \theta = 2.7 \), for out-degree. In [6], a single subset of Web Graph is analyzed - the Web of a single country (Web of Spain) and similar distribution is observed, with \( \theta = 2.11 \) for in-degree and \( \theta = 2.84 \) for out-degree, validating the scale-free structure of the Web Graph and indicating that the link distribution is invariant to the change of scale (we use this idea in proposing the concept of query-induced subgraphs). Complete statistical analysis of topic-related link graphs, generated in social networks, is given in [7]. Authors observe the power law distribution of node degrees and propose power law based on truncated-log-normal hypothesis. Finally, paper [8] gives a complete description of methods for estimating power-law distribution parameters from empirical data.

General overview of graph clustering algorithms and appropriate metrics for determining cluster quality is given in [9]. Efficient graph clustering algorithms vary in computational complexity from \( O(n^3) \), in the case of recursive partitioning, to \( O(n \log n) \) in the case of multilevel clustering algorithm described in [10]. However, all of given algorithms operate in \( O(n^2) \) space complexity, as they require availability of entire graph representation, making it hard for implementation on the scale of \( n \) found in practical problems.
III. QUERY-INDUCED SUBGRAPHS

A. Definition

Let the hyperlink graph be a graph G = (V, E), where V is a set of vertices representing all the documents in the search engine index, and E is a set of edges representing hyperlinks between all the documents.

We define Query-Induced Subgraph as a graph G_q = (V_q, E_q), where V_q ⊂ V is a set of all results matching the given query q and E_q ⊂ E set of all edges between vertices from the set V_q. In practice, given subgraph (G_q ⊂ G) represents the hyperlink graph created from G by keeping only the documents matching given query and hyperlinks between the documents in resulting set. We define node degree as number of links (both inlinks and outlinks) for each node, and treat it as a measure of information content contained in link data. Our goal is to show that the node degree in the given query-induced subgraph, preserves the same distribution as in the entire graph (anticipated by the general assumption about scale-free structure of Web and social networks [5]).

B. Properties

In order to validate the given assumption about degree distributions, we analyze the dataset obtained as a part of randomNode clustering engine. Given dataset consists of data about 1.1 million nodes (representing the subset of .yu Web), generated by calculating inlink degrees for resulting sets of 1000 top-frequency queries in randomNode clustering engine. We analyze the distribution of inlink degrees for both full graph and induced subgraphs obtained for each of given queries and test the hypothesis that both graphs have distribution, commonly found in Internet and social networks [7] - a power law distribution with β and x_min parameters, and probability density function of the form:

\[ p(x; \beta, x_{min}) = \frac{x^{-\beta}}{\zeta(\beta, x_{min})} \]

where \( \zeta(\beta, x_{min}) \) represents the generalized zeta function

\[ \zeta(\beta, x_{min}) = \sum_{n=0}^{\infty} n^{(n + x_{min})} \]

We use the method of Maximum Likelihood (ML) for estimation of distribution parameters, as described in [8]. The approximate expression for MLE estimator of \( \beta \) parameter is given by:

\[ \hat{\beta} \equiv 1 + n \left[ \sum_{i=1}^{\infty} \ln \frac{x_i}{x_{min} - \frac{1}{2}} \right]^{-1} \]

Finally, from Table I we observe estimated values of \( \beta = 2.500576 \) for full graph and \( \beta = 2.533536 \) for induced subgraph, which validates the proposed concept of scale-invariance of graph structure. This further indicates that the essential graph properties (high-degree "authoritative" nodes [3] and random walk convergence properties [4]), existing in the entire graph, are still preserved in the query-induced subgraph. Hence, we can reduce the dimension of search result clustering problem, by restating it as a problem of clustering the query-induced subgraph \( G_q \), corresponding to the given query \( q \). Such problem relaxation enables us to perform computation in much efficient manner, while still preserving essential information contained in the link structure.

IV. ALGORITHM FOR FAST CLUSTERING USING RANDOM WALKS ON POWER-LAW GRAPHS

A. Description

We propose an algorithm for graph clustering using random walks on directed power-law graphs. The algorithm operates by performing a number of independent random walks on the link graph and attempts to exploit the specific structure of common power-law graphs in order to bound the average walk length. For each walk, we record a number of times each node was visited, and obtain partial sets, each containing the nodes visited during the walk and appropriate visit counts. Finally, we use that info in order to perform the merge stage of the algorithm, in which we use pivot nodes (nodes with maximum visit counts), in order to merge the given partial sets into a number of final sets, representing the cluster set for a given graph.

B. Algorithm

Let the \( G(V, E) \) be the connected, directed graph with \( |V| = N \) and \( |E| = m \). By random walk on graph, we assume Markov chain \( M_q \), where \( V \) represents the set of states of the chain and \( \bar{P} = [p_{ij}] \) is a stochastic matrix,
with \( p_{ij} \) representing transitional probability for any two states \( i, j \in V \), given by:

\[
P_{ij} = \begin{cases} 
\frac{1}{\deg_i}, & \text{if } \exists (i,j) \in E \\
0, & \text{if otherwise}
\end{cases}
\]

and \( \deg(i) \) represents the outdegree of a vertex \( i \).

We define the stationary distribution of a Markov chain \( M_g \) corresponding to a given walk on graph \( G \), as a probability distribution \( \pi \), such that \( \pi = \pi \cdot P \), were each entry \( \pi_i \) is proportional to the amount of time walk will spend in a given node. Such distribution is often used as a measure of importance of given node \( i \). In the undirected case, the random walk on the graph converges to the stationary distribution [1], as well as in the case of directed strongly connected graph [12]. Although this does not hold for the distribution [1], as well as in the case of directed strongly connected component of the random walk on directed graph as a state case of power law graphs [11]. Additionally, we define the strongly connected components of such graph, which are shown to exist in the general case of power law graphs [11].

### Algorithm 1 Random Walk Clustering

**Require:** Graph \( G(V,E) \) and approximation factor \( K \), where \( |V| = N \), \( k \in [0,1] \) and \( K = k \cdot N \)

**WALK phase:**

\[
i \leftarrow 0
\]

while \( i \leq K \) do

\[
s \leftarrow \text{rand}(1,N)
\]

while \( s \neq 0 \) do

if \( s = \{i \} \) then

\[
w_i \leftarrow (s,1)
\]

end if

\[
s(w_i) \leftarrow s(w_i) + 1
\]

\[
s \leftarrow \text{rand}(\text{adj}(s)); v \in \text{adj}[s \rightarrow v] \in E
\]

if \( \deg(s) > \deg(n) \) then

\[
s \leftarrow 0
\]

end if

end while

we get the walk set \( W = \{w_1,...w_k\} \)

**MERGE phase:**

for each \( w_i \in W, i \in (1,K) \):

for each node \( n \in w_i \):

if \( \exists w_k[(\deg(s) - \deg(n)] > T_{cm} \) then

we remove (cut) node \( n \) from \( w_i \)

end if

if \( \exists w_k[(\deg(s) - \deg(n)] < T_{cm} \) then

we perform merge of \( w_i \) and \( w_m \)

end if

return \( C = \{w_1,...w_m\}, m \leq K \) - the final set of clusters in given graph

For the purpose of a given algorithm, we define stopping condition for given walk either as a condition of process entering the stopping state, or as a threshold value for the length of the walk. Due to the nature of the underlying graph, not every walk will enter the stopping state, since the loops might occur, therefore we must define additional maximum walk length \( L \) (usually of \( O(N) \) order), which should prevent infinite loops, yet be large enough for the walk to capture the sufficient approximation of a distribution of node visit counts for given walk.

We perform the WALK phase of the algorithm by selecting \( K = k \cdot N \) random nodes, where \( k \in (0,1) \), represents the approximation constant of the algorithm, and performing \( K \) walks on graph \( G \). Walks are performed until they reach the stopping condition, either by entering the stopping state or by hitting the maximum walk length.

Finally, in the MERGE size, we sort walks by length, and internally by visit count, and iterate the result set by performing CUT and MERGE operations, interchangeably. If, for a given node, there is a walk having visit count significantly greater than in the current walk, we remove it (CUT) from given walk, whereas, if there is a walk having similar visit count for a given walk, we perform MERGE of two walks based on given (pivot) node. In such manner, we hope to identify the key (pivot) nodes for every walk, and perform a join of two walks in case they share the key nodes. Additionally, by manipulating the threshold value for cut/merge \( (T_{cm}) \), we can efficiently manipulate the dimension of the clustering, balancing between cluster number and cluster size.

### C. Analysis

In order to analyze given algorithm, we use results proved in [11], stating that for a class of power law graphs with \( N \) nodes and exponents in range \( \beta \in (2,3) \) (which correspond to the general case of Internet, social and citation networks, such as the dataset analyzed in this paper), average distance between any two is almost surely of order \( O(\log \log(N)) \). In such a graph, it is guaranteed that there are more than zero terminating nodes, and the expected average distance between arbitrary node and given terminating node is of order \( O(\log \log(N)) \). Therefore, we can determine that the expected average run length of the WALK phase is of the \( O(N \log \log(n)) \) order. Additionally, such graphs contain the strongly connected component of the size \( n^{c/\log \log(n)} \) [11], therefore, we define the \( O(N) \) maximum walk length in order to cover walks not hitting the terminating node. This finally results in \( O(N^2) \) worst case time for a given algorithm and \( O(N \log \log(N)) \) expected average case time for the WALK phase and for a complete algorithm (the merge phase can be implemented efficiently in \( O(N \log \log(N)) \) time).

However, although the worst case time of given algorithm is \( O(N^2) \), both his average running time, and the fact that by reducing the problem to the induced subgraph, we operate on \( N \) which represents the number of nodes matching the given query and is significantly smaller than the total number of nodes in search engine index. Additionally, given random walk implementation is much more space efficient, as it only requires storage of adjacency list for every node (of \( O(N \log(N)) \) order) to perform random walks and get partial sets, as opposed to the matrix-based eigenvalue methods, which require \( O(N^2) \) space for storage of the entire adjacency matrix.
V. Results

As a part of the research, and as a base for obtaining practical results, we have created a clustering search engine called \textit{RandomNode}, accessible at http://www.randomnode.com, which performs query-time clustering of search results by implementing the \textit{Random Walk Clustering} algorithm, proposed in section IV, implemented on top of the \textit{Lucene} search library. It operates on 1.1-million node dataset, represents a significant portion of .yu web, generated by performing a crawl starting at the homepage of the Belgrade University (http://www.bg.ac.yu).

We use the \textit{randomNode} clustering engine in order to analyze the impact of approximation factor $K$ on the performance of the proposed algorithm. We use the coverage($C$), of a graph clustering $C$ = ($C_1$...$C_k$), as a measure of clustering quality, defined as:

$$\text{coverage}(C) = \frac{m(C)}{m} = \frac{m(C)}{m(C) + \hat{m}(C)}$$

where $m(C)$ represents the number of inter-cluster edges, while $\hat{m}(C)$ represents a number of intra-cluster edges. Optimal clustering should minimize the $\hat{m}(C)$, as it represents the size of the cut in the graph performed by given clustering.

We perform analysis using \textit{randomNode} engine, by performing clustering on 1000 top-scoring keywords in given dataset, varying the approximation coefficient in the (0.1, 1.0) range with 0.1 step and calculating the coverage metric. The results are shown in Figure III, with scatterplot showing exact coverage values for each of each sample instance and the average coverage, given by the line segment. We observe that the coverage increases logarithmically with the approximation coefficient, which indicates that the algorithm can provide acceptable approximations, even for the small values of $K$. Finally, we use the randomNode engine to extract a set of queries, shown in Table II, representing top-scoring clusters, both in terms of results and a cluster coverage, for a given subset of .yu Web.

We observe that the coverage increases logarithmically with the approximation coefficient, which indicates that the algorithm can provide acceptable approximations, even for the small values of $K$. Finally, we use the randomNode engine to extract a set of queries, shown in Table II, representing top-scoring clusters, both in terms of results and a cluster coverage, for a given subset of .yu Web.

\begin{table}[h]
\centering
\caption{Top clusters in randomNode dataset}
\begin{tabular}{lcccc}
\hline
query & coverage & n.links & incluster & n.clusters & max size \\
\hline
politika & 0.999 & 37475 & 37417 & 29 & 820 \\
pravda & 0.967 & 34688 & 33556 & 43 & 682 \\
rubrike & 0.995 & 33200 & 33053 & 13 & 817 \\
shop & 0.967 & 29440 & 28482 & 88 & 549 \\
nekretnine & 0.988 & 28451 & 28157 & 30 & 535 \\
leasing & 0.988 & 28185 & 27847 & 35 & 272 \\
dekanat & 0.947 & 27873 & 27264 & 63 & 326 \\
banking & 0.965 & 26840 & 25916 & 120 & 211 \\
expo & 0.963 & 26456 & 24629 & 88 & 549 \\
filologija & 0.976 & 23160 & 22609 & 39 & 625 \\
\hline
\end{tabular}
\end{table}

\section*{Acknowledgment}

Thanks to prof. Veljko Milutinovic, who mentored this research as a part of my diploma thesis at the faculty of Electrical Engineering, Belgrade.

\section*{References}

[1] G. Pandurangan, P. Raghavan and E. Upfal, \textit{Using PageRank to Characterize Web Structure} Internet Mathematics Volumes 3, Number 1, 2006

[2] M. Hearst, and J. Pedersen, Reexaming the cluster hypothesis: scatter/gather on retrieval results Proceedings of the 19th annual international ACM SIGIR conference on Research and development in information retrieval, 1996

[3] L. Li and Y. Shang, Improvement of HITS-based Algorithms on Web Documents Proceedings of the 11th International World Wide Web Conference, 2002

[4] J. Huang, T. Zhu and D. Schuurmans, Web Communities Identification from Random Walks Lecture Notes in Computer Science, Knowledge Discovery in Databases: PKDD 2006, Volume 4213, 2006

[5] A. Broder, R. Kumar, F. Maghoul, P. Raghavan, S. Rajagopalan, R. Stata, A. Tomkins, and J. Wiener, Graph structure in the web : Experiments and models Proceedings of the Ninth Conference on World Wide Web, pages 309320, 2000

[6] R. Baeza-Yates, C. Castillo and V. Lopez, Characteristics of the Web of Spain Cybermetrics, Vol. 9, No. 1, 2005

[7] V. Gomez, A. Kaltenbrunner and V. Lopez, Statistical analysis of the social network and discussion threads in slashdot Proceeding of the 17th international conference on World Wide Web, 2008

[8] A. Clauset, C.R. Shalizi, and M.E.J. Newman, Power-law distributions in empirical data E-print (2007). arXiv:0706.1062

[9] U. Be, M. Gaertler and D. Wagner, Experiments on graph clustering algorithms Lecture notes in computer science, Springer-Verlag, 568-579, 2003

[10] H. Djdjev, A scalable multilevel algorithm of graph clustering and community structure detection Fourth Workshop on Algorithms and Models for the Web-Graph, 2006

[11] F. Chung and L. Lu, \textit{The Average Distance in a Random Graph with Given Expected Degrees} Internet Mathematics Vol. I, No. I : 91-114, 2002.

[12] F. Chung, L. Lu, and V. Vu \textit{The Spectra of Random Graphs With Given Expected Degree} Proceedings of National Academy of Sciences, 100:6313-6318, 2003