ISSN No. 0976-5697

Volume 5, No. 1, Jan-Feb 2014

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

# On Enumerating Maximal Bicliques of Click-Through Graph

G.Manoranjitham
Department of Computer Science and Engineering
Kalaignar Karunanidhi Institute of Technology
Coimbatore, India
manorangitham1991@gmail.com

A.Revathi
Department of Computer Science and Engineering
Kathir College of Engineering
Coimbatore, India
revs.a1990@gmail.com

***Abstract:*** Search Engine is to retrieve the relevant information from the web, based on user queries. The problem of discovering query clusters from a click-through graph of web search logs is described in this paper. The click through graph consists of a set of clicked queries, pages that are returned as result and a set of edges that connects a query and a page clicked by the user. The proposed method is enumerating all maximal bicliques of a bipartite graph from click-through graph and compute equivalence query clusters from maximal bicliques. It has two main contributions: first, use all maximal bicliques to find minimum number of bicliques covering a subset of edges or vertices for given bipartite graph. We have conducted experiments on Yahoo web search queries.

***Keywords:*** biclique enumeration; click-through graph; query cluster;

## I. INTRODUCTION

A search engine can track which of its search results were clicked for which query. For a popular system, these click records can amount to millions of query-document pairs per day. Users select pages by clicking on links on a search engine result page that are related to queries. The query clustering method includes query and clicked page relationship, not considering syntactic or semantic features on the query, such as keywords.

From the perspective of a user conducting a search, documents that are clicked but not relevant constitute noise in the click data. Documents that are relevant but not clicked constitute sparsity in the click data. One class of approaches attempts to reduce noise in click data, by building a click model that may use additional information about the user's behaviour. For example, taking into account the user's browsing patterns after clicking a document. These approaches can significantly reduce noise, by identifying some clicked documents as irrelevant.

In this paper, we investigate efficient algorithm for finding all maximal bicliques in general bipartite graphs. A bipartite graph $\beta = (X, Y, \varepsilon)$ is a graph, were the vertices can be divided into two disjoint sets X and Y such that every edge $e_{ij} \epsilon\, \varepsilon$ connects a vertex in X to one in Y. A biclique $(S_x, S_y)$ of a bipartite graph β is a complete bipartite sub graph of β induced by vertex set $S_x \cup S_y$ . The neighbor set $N_y(x_i)$ of a vertex $x_i \epsilon\, X$ is defined as the set of vertices $y_i$ such that there is an edge $e_{ij} \epsilon\, \varepsilon$, i.e., $N_y(x_i) = \{y_i \epsilon\, Y : (x_i, y_i) \epsilon\, \varepsilon\}$. Similarly, $N_x(y_j) = \{x_i \epsilon\, X : (x_i, y_j) \epsilon \varepsilon\}$. The empty biclique $(\emptyset, \emptyset)$ is simply denoted as Ø.

## II. MAXIMAL BICLIQUE ENUMERATION

Definition 1: (Maximal Biclique) Given a bipartite graph $\beta = (X, Y, \varepsilon)$, a biclique $(S_x, S_y)$ is a maximal biclique of β if no proper superset of $(S_x, S_y)$ is a biclique, i.e., there exists no biclique $(S'_x, S'_y) \neq (S_x, S_y)$ such that $S_x \subseteq S'_x$ and $S_y \subseteq S'_y$.

For the sake of simplicity, pair of sets $(S_x, \{y_j\})$ and $(\{x_i\}, S_y)$ are denoted as $(S_x, y_j)$ and $(x_i, S_y)$, respectively [1].

Definition 2: (Consensus Set) Given a bipartite graph $\beta = (X, Y, \varepsilon)$, for a subset $S_x \subseteq X (S_Y \subseteq Y)$ the consensus set $P_y(S_x)(P_x(S_y))$ is defined as the intersection of neighbor set of each vertex $x_i \in S_x$ $(y_j \in S_y)$, i.e.,

$$P_y(S_x) = \cap\ N_y(x_i) \qquad (1)$$

By definition, $P_x(\emptyset) = P_y(\emptyset) = \emptyset$.

Note that this is equivalent to $P_y(S_x) = \{y_j \epsilon\, Y : (S_x, y_j) is\ biclique\}$ and similarity $P_x(S_y) = \{x_i \epsilon X : (x_i, S_y) is\ biclique\}$ [1].

Theorem 1: $(S_x, S_y) = \emptyset$ is a maximal biclique$\Leftrightarrow$ $S_y = P_y(S_x)$ and $S_x = x(S_y)$.

*Proof:* Let $(S_x, S_y) \neq \emptyset$ be a biclique and so $S_x \subseteq P_x(S_y)$ and $S_y \subseteq P_y(S_x)$. By definition of maximality, $S_x, S_y$ is a maximal biclique if and only if there exists no $y_j \in Y/S_y$ such that $(S_x, y_j)$ is a biclique and similarly there exists no $x_i \in X/S_x$ such that $(x_i, S_y)$ is biclique. This is equivalent to that there exists no $y_j \in Y/S_y$ such that $y_j \in P_y(S_x)$ and there exists no $x_i \in X/S_x$ such that $x_i \in P_x(S_y)$. Equivalently $S_y = P_y(S_x)$ and $S_x = P_x(S_y)$.

Lemma 1 : For any $S_y \subseteq Y$, $S_y \subseteq P_y(P_x(S_y))$. Similarly, for any $S_x \subseteq X$ , $S_x \subseteq P_x(P_y(S_x))$.

*Proof:* Consider a vertex $y_j \in S_y$. Since $(P_x(S_y), S_y)$ is biclique, $(P_x(S_y), y_j)$ is also biclique. Thus, $y_j \in P_y(P_x(S_y))$ this concludes that $S_y \subseteq P_y(P_x(S_y))$. Similar proof can be conducted for $S_x \subseteq P_x(P_y(S_x))$.

Theorem 2: For any $S_y \subseteq Y$ such that $P_x(S_y) \neq \emptyset$, $(P_x(S_y), P_y\left(P_x(S_y)\right))$ is a maximal biclique. Similarly, for any $S_x \subseteq X$ such that $P_y(S_x) \neq \emptyset$, $(P_x\left(P_y(S_x)\right), P_y(S_x))$ is a maximal biclique.

*Proof:* Let $S_y$ be a subset of Y such that $P_x(S_y)$ and let $\overrightarrow{S_y}$ denote $P_y(P_x(S_y))$. Consider a vertex $x_i \in P_x(\overrightarrow{S_y})$. Then, $(x_i, S_y)$ is biclique. Since $S_y \subseteq S_y$, $(x_i, S_y)$ is also biclique.

Thus $x_i \in P_x(S_y)$ which concludes that $P_x(S_y) \supseteq P_x(S_y)$. Since $P_x(S_y) \subseteq P_x(\overrightarrow{S_y})$, $P_x(S_y) = P_x(\overrightarrow{S_y})$ Since both $P_y(P_x(S_y)) = S_y$ and $P_x(S_y) = P_x(S_y)$, $(P_x(S_y), \overrightarrow{S_y})$ is a maximal biclique. Similar proof can be conducted for maximality of biclique $(P_x\left(P_y(S_x)\right), P_y(S_x))$ [1].

## III. DATASET AND PREPROCESSING

In this study we used a set of randomly sampled click-through data from the Yahoo web search query logs. The sample data consists of over 16M unique queries, over 10M page URLs, and over 92M edges connecting the nodes [2].

Like a typical web graph, the click-through graph exhibits the power law distribution in terms of the out-degree of query nodes, and the in-degree of clicked URLs. We prune the following nodes and their associated edges in the preprocessing step:

a. Web pages with in-degree higher than 100 and their in-coming edges: these are pages of very broad topic and interest, such as www.craigslist.org, and en.wikipedia.org. With those types of pages, we would get too broad clusters, since the topics of queries connected to those pages tend to be vary. There are about 5% of unique URLs with this level of high in-degrees.
b. Queries with out-degree greater than 10 and their out-going edges: most normal user queries result in only a small number of clicks, rarely more than 10 pages. Those with high out degree may include ones by robots for scraping or some special types of queries. There are only about 0.1% of URL queries in this category.
c. Web pages with in-degree 1 and their in-coming edge.
d. Queries with out-degree 1 and their out-going edge.
e. Edges with click frequency less than a threshold $\tau$.

## IV. QUERY CLUSTERING

### A. *Biclique Generation:*

[3] Shows maximal bicliques generation from a bipartite graph is a special case of the maximal clique generation problem from a general graph. Let $G = (V_1 \cup V_2, E)$ be a bipartite graph, where $V_1$ and $V_2$ are the two disjoint sets of nodes, and $E$ is a set of edges connecting nodes in $V_1$ and $V_2$. To generate maximal bicliques, $G$ is transformed to a general graph $G' = (V_1 \cup V_2, E')$ where $E' = E \cup (V_1 X V_1) \cup (V_2 X V_2)$. Then the maximal clique generation algorithm for a general graph, such as the one in [4], can be applied on $G'$. This algorithm, however, requires an increased amount of the main memory space proportional to the entire expanded graph.

We deal with a very large click-through graph that would hardly fit into main memory. We instead modified the bipartite core generation algorithm in [1] to generate maximal bicliques. One of the major advantages of the algorithm is that it does not require storing the entire graph in memory. Instead it applies various pruning techniques on sorted lists of nodes; one for queries, and another for pages for click-through graph. The algorithm can be further optimized to sort only once, and build only a small index in main memory [2].

#### a) *Iterative pruning:*

Since we are looking for query clusters larger than certain size, queries and pages of which in- and out-degrees do not meet the minimum size requirement can be eliminated.

To compute a biclique of size ($i$,$j$), query nodes with out-degree smaller than $i$ and their out-going edges are pruned. Similarly, any page node with in-degree smaller than $j$ and its associated edges are pruned. This pruning step is iteratively applied until there exists no more such nodes.

#### b) *Biclique generation:*

At each step of the biclique generation algorithm we either generate a biclique, or exclude a node and the associated edges from the graph. After generating a biclique, the sub graph corresponding to the biclique is removed from the click-through graph. Starting from the maximum out-degree size, we repeat the following steps iteratively for each decreasing value of $i$:

(a). From a sorted list of queries, find all queries, $q_k$, without degree $i$, and list the neighbors of each $q_k, P(q_k)$.
(b). For each $P(q_k)$, generate the set of all in-coming queries, $Q(P_l)$, of each page, $P_l \in P(q_k)$. (An index on the page URLs is used for better performance.)
(c). Find the intersection of all $Q(P_l)$;$\cap_{l \in P(q_k)} Q(P_l)$.
(d). Let $m$ denote the size of the query set, $\cap_{l \in P(q_k)} Q(P_l)$, and $E$ be a set of all edges between $\cap_{l \in P(q_k)} Q(P_l)$ and $P(q_k)$.

i. If ($m \geq j$), generate a biclique of size ($i$,$m$), $(\cap_{l \in P(q_k)} Q(P_l) \cup P(q_k), E)$. After generating a biclique, remove all query and page nodes, and edges in the biclique from the click-through graph (unless the node is a part of another cluster).
ii. If ($m < j$), remove all in-coming edges of nodes in $P(q_k)$ that do not connect to a query node in the intersection $\cap_{l \in P(q_k)} Q(P_l)$.

(e). After removing all edges, apply the iterative pruning before continuing with the next iteration with out-degree size $i$-1.

The generated bicliques are maximal.

### B. *Query Cluster Generation:*

For each biclique generated, the query set, $\cap_{l \in P(q_k)} Q(P_l)$ forms an equivalence set that becomes a query cluster.

## V. ALGORITHM

Theoretical results suggest that it is sufficient to enumerate on the consensus sets, in order to find all maximal bicliques [1]. Therefore we find all consensuses X subsets of bipartite graph $\beta = (X, Y, \varepsilon)$. The algorithm requires only the bipartite graph B. Then we initialize a set of consensus X subsets S with neighbor sets of every vertex $y_j \in Y$. Concurrently, we hold a priority queue Q which works in a FIFO manner, and it is also initialized by the same set of consensus X subsets. We iteratively grow the set S as follows. At each iteration, we select an unselected consensus set $S_x$ from queue Q. For each vertex $y_j \in Y$ which is not in the consensus set of $S_x$, we construct a set

$S_{new}$ by the intersection of $S_x$ and neighbor set N($Sy_j$). $S_{new}$ corresponds to the consensus set of $P_y(S_x) \cup \{y_j\}$. We do not consider a vertex in $P_y(S_x)$, because for such a vertex, the intersection will result again $S_x$ which wouldn't be a new consensus set in S . If $S_{new}$ is a new consensus set in S , we insert $S_{new}$ to S . We also enqueue it to priority queue Q in order to expand new consensus sets based on $S_{new}$. The iterations terminate whenever there remains no consensus set to generate new ones. By the termination, we compute the maximal bicliques by taking pairs $(S_x, P_y(S_x))$ for each subset $S_x \in S$.

Algorithm 1 FIND-ALL-MAXIMAL Algorithm

Require: Bipartite graph $\beta = (X, Y, \varepsilon)$
S ← $\{N_x(y_j) : y_j \in Y\}$
Q ← S
While $Q \neq \emptyset$ DO
$S_x$ ← DEQUEUE(Q)
for each $y_j \in Y / P_y(S_x)$ do
$S_{new} \leftarrow S_x \cap N(y_j)$
if $S_{new} \notin S$ then
$S \leftarrow S \cup \{S_{new}\}$
ENQUEUE(Q, $S_{new}$)
end if
end for
end while
$C_{max} = \{(S_x, P_y(S_x)) : S_x \in S\}$
return $C_{max}$

For each maximal biclique we check for at most |Y | new bicliques. With a naïve implementation, the checking procedure can be done in polynomial time on number of total maximal bicliques and number of vertices. Thus, the whole procedure runs in polynomial-time in total of input and output size with a naive implementation which concludes that FIND-ALL-MAXIMAL is a *total polynomial* algorithm [4] for the problem of enumerating all maximal bicliques of a bipartite graph [1].

## VI. EXPERIMENTS

After preprocessing with τ=2, there remain ~1.15M and ~2M query and page nodes, and ~68M edges in the sample graph, reduced from ~16M and ~10M nodes and ~ 92M edges, respectively [2].

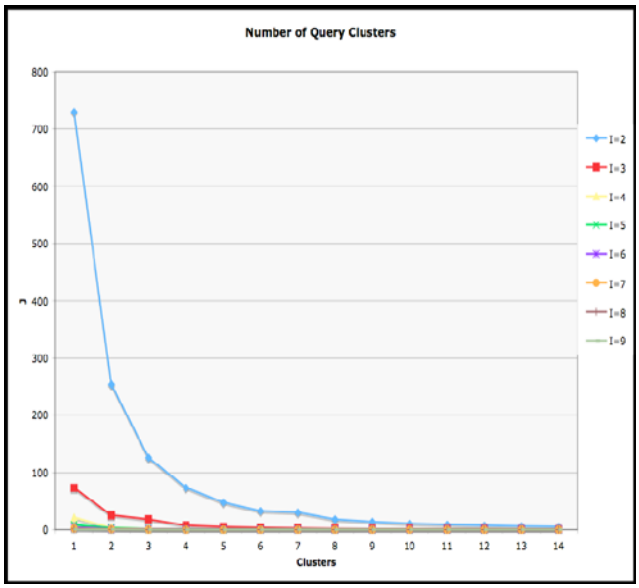


Figure 1: Number of query clusters

Figure 1 plots the number of query clusters extracted by our algorithm. As expected, the numbers of maximal bicliques drop significantly as the size of the cliques grow. The number may be interpreted as lower bound of query clusters, as our method considers only maximal bicliques. As we relax the equivalence condition and consider strongly connected, but not necessarily completely connected bipartite sub graphs as the candidates, it may further reveal interesting quasi-equivalence sets of queries. Due to the strict requirement of complete connectedness of the clusters by the current algorithm, many potentially interesting query clusters are excluded if they slightly violate the requirements.

## VII. REFERENCES

[1] Enver Kayaaslan, "On Enumerating All Maximal Bicliques of Bipartite Graphs" CTW2010, University of Cologne, Germany. May 25-27, 2010.

[2] J. Yi and F. Maghoul, "Query Clustering Using Click-through Graph," Proc. the 18th Int'l Conf. World Wide Web (WWW '09), 2009.

[3] K. Makino, and T. Uno, New algorithms for enumerating all maximal cliques, The 9th Scandinavian Workshop on Algorithm Theory, 2004.

[4] E. Tomita, A. Tanaka, and H. Takahashi. The worst-case time complexity for generating all maximal cliques and computational experiments. Theoretical Computer Science, 363(1), pp.28-42, 2006.

[5] G. Alexe, S. Alexe, Y. Crama, S. Foldes, P. L. Hammer, and B. Simeone, Consensus algorithms for the generation of all maximal bicliques, Discrete Applied Mathematics, 145 (2004), pp. 11 – 21. Graph Optimization IV.

[6] J. Orlin, Contentment in graph theory: Covering graphs with cliques, Proceedings of the Koninklijke Nederlandse, (1977), pp. 406–424.