Efficient Processing of Group-Oriented Connection Queries in a Large Graph

James Cheng
School of Computer Engineering
Nanyang Technological University, Singapore
jamescheng@ntu.edu.sg

Yiping Ke
Department of Systems Engineering and Engineering Management
Chinese University of Hong Kong, Hong Kong
ypke@se.cuhk.edu.hk

Wilfred Ng
Department of Computer Science and Engineering
Hong Kong University of Science and Technology, Hong Kong
wilfred@cse.ust.hk

ABSTRACT

We study query processing in large graphs that are fundamental data model underpinning various social networks and Web structures. Given a set of query nodes, we aim to find the groups which the query nodes belong to, as well as the best connection among the groups. Such a query is useful to many applications but the query processing is extremely costly. We define a new notion of *Correlation Group* (CG), which is a set of nodes that are strongly correlated in a large graph G. We then extract the subgraph from G that gives the best connection for the nodes in a CG. To facilitate query processing, we develop an efficient index built upon the CGs. Our experiments show that the CGs are meaningful as groups and importantly, the meaningfulness of the query results are justifiable. We also demonstrate the high efficiency of CG computation, index construction and query processing.

Categories and Subject Descriptors: H.2.8 [Database Management]: Database Applications - Data Mining

General Terms: Algorithms

Keywords: Connection query, correlated group, social networks

1. INTRODUCTION

Graph is known as the most general data model for representing and understanding objects and their relationships in various application domains. In recent years, graph databases have become more in use and the volume of graph data increases rapidly. Thus, it is important to develop efficient algorithms for processing queries in graph databases.

Let $\mathcal{W} = (V_\mathcal{W}, E_\mathcal{W})$ be a graph, where $V_\mathcal{W}$ and $E_\mathcal{W}$ are the set of nodes and edges in $\mathcal{W}$. The nodes in $V_\mathcal{W}$ are uniquely labeled and the edges in $E_\mathcal{W}$ are undirected and weighted. We call $\mathcal{W}$ a *uniquely-labeled weighted graph*, or simply a *graph* in this paper. Given a set of query nodes, Q, the query processing problem we study in this paper is described as follows:

1. *Find the* group *to which each query node $q \in Q$ belongs, where the group of q is a set of nodes in $\mathcal{W}$ that are mostly correlated to q as measured by a correlation score.*
2. *For the group of each $q \in Q$, extract a subgraph from $\mathcal{W}$ that gives the best connection for the nodes in the group by maximizing a correlation score.*
3. *Compute the best connection among the groups of all $q \in Q$ in $\mathcal{W}$, which is the answer to the query.*

We name such a query a *group-oriented connection* query. Group-oriented connection queries are important to a wide range of applications. For example, how organizations and people do business together, how terrorists plan and conduct terrorist activities together and how the different terrorist groups are related to each other, and how the various social networks behave. It can also be applied to other domains such as gene regulatory networks, viral marketing, the Internet, and many more.

One of the challenges in processing group-oriented connection queries is the formation of groups for different query nodes. Consider two nodes in the graph, how do we decide whether they are in the same group or not? A common way is to consider every maximal clique as a group. However, in many cases we cannot simply consider only the edge connection, especially when some edges between the nodes in the clique may have very low edge weight while others have very high edge weights. Therefore, the nodes that belong to the same clique may not really mean that they have close connection with all other nodes. On the contrary, many other structures, such as a star or a ring, may well represent a significant relationship between the nodes and can be considered as a group.

In literature, the *proximity graph* [4] and the *center-piece subgraph* ($CEPS$) [8] may be considered as a group. However, the computation of the proximity graph and the CEPS is costly in general, especially for a large graph. Community detection [7, 10, 3, 6, 1] can be employed to define groups in a graph as communities; however, communities are usually disjoint with each other, whilst in a large graph such as a social network, many people may belong to different groups at the same time.

CIKM'09, November 2–6, 2009, Hong Kong, China.

We define groups by considering the relation between nodes in a graph. The relation of one node u to another node v is measured by a *relevance score* based on the concept of *Random Walk with Restart* (*RWR*) [9]. There are many ways to define the relevance measure, such as the shortest path and the maximum flow. However, these measures are shown to be inadequate in capturing the relation between two nodes [2]. On the other hand, RWR is robust to various graph structures and has been shown to be able to capture the relationship between people in social networks [8, 9].

Based on the node relation, we define group, called a *Correlation Group* (*CG*), as the set of nodes that are pair-wise related with each other. To avoid redundancy, we require a CG to be the maximum set of nodes that are correlated. For each CG, we also extract a subgraph, called a *Correlation subgraph* (*C-graph*), that reflects the correlation or relation among the nodes.

To facilitate query processing, we develop an *inverted-index* built upon the set of CGs. We devise efficient algorithms for both the index construction and the query processing. Our extensive experiments demonstrate the meaningfulness of the CGs and the C-graphs, which are shown to indeed capture the relation among the nodes as a group. The results also show the efficiency of the index construction and query processing. We also justify the meaningfulness of our query answer.

2. CORRELATION GROUPS

We start by defining the notion of a *group* in a graph $\mathcal{W}$ (See Section 1 for the definitions of a graph and our problem). Semantically, a group refers to a set of objects that have a certain "*group*" relation with each other such that they are regarded as belonging together. We first define this "*group*" relation.

We define *correlation* between two nodes as follows.

Definition 1 (Correlation). *Let u and v be two nodes in $\mathcal{W}$. Let $r(u,v)$ be the* relevance score *from u to v. Let σ be a predefined* minimum relevance threshold. *Then, u is* related *to v if $r(u,v) \geq \sigma$; u and v are* correlated *if $r(u,v) \geq \sigma$ and $r(v,u) \geq \sigma$.*

The relevance score $r(u,v)$ in Definition 1 is a measure to quantify the degree of the relation from u to v. We can adopt a specific measure to suit the need of a specific application. For general purpose, in this paper we use *Random Walk with Restart* to define the relevance score, for its robustness to various graph structures.

After defining the correlation between two nodes, we can now extend the correlation to a set of nodes as follows.

Definition 2 (Correlated Nodeset). *Let U be a set of nodes in $\mathcal{W}$. U is a* Correlated Nodeset *(*CN*) if $\forall u, v \in U$, $r(u,v) \geq \sigma$.*

It follows from Definition 2 that every node in a CN must be correlated to all nodes (including itself) in the CN. However, the definition of CN reveals that every subset of a CN is also a CN. If we define a group to be a CN, then there are exponentially many groups, the majority of which share many common nodes. To address this problem, we define a group to be a maximal CN as follows.

Definition 3 (Correlation Group). *Let U be a set of nodes in $\mathcal{W}$. U is a* maximal CN *if U is a CN and $\nexists U' \supset U$ such that U' is a CN. U is said to be a* Correlation Group *(*CG*) if and only if U is a maximal CN. A CG U is assigned a* correlation score*, $r(U)$, as given by the following equation:*

$$r(U) = MIN\{r(u,v) : u, v \in U\}.$$

For computing the CGs, we only require those relevance scores that are no less than σ. We define a graph G_{score} based on the relevance scores as follows. The set of nodes in G_{score} is the set of nodes in $\mathcal{W}$. For any two nodes u and v in G_{score}, there is an edge (u,v) in G_{score} if $r(u,v) \geq \sigma$ and $r(v,u) \geq \sigma$. It is not difficult to see that the set of CGs is just the set of maximal cliques in G_{score}. We can then apply any existing algorithm for computing the set of CGs.

3. INDEX CONSTRUCTION

In this section, we devise an index for processing group-oriented connection queries.

First, given the set of CGs, $\mathcal{C}_G$, we construct an inverted-index on $\mathcal{C}_G$. We define the structure of the inverted-index as follows.

Definition 4 (CG-Index). *An inverted-index constructed on $\mathcal{C}_G$, called the* CG-index*, consists of the following components:*

- CG Arrays (CGA)*: a two dimensional array which stores $\mathcal{C}_G$, where each array CGA[i] stores the set of CGs, S, where each CG in S consists of exactly i nodes. The CGs in each CGA[i] are sorted in descending order of their correlation scores.*
- Node Array (NA)*: an array stores the set of nodes, $\bigcup_{U \in \mathcal{C}_G} U$.*
- ID-array*: for each distinct node u in the NA, we associate with u the set of IDs of the CGs that contain u. The set of IDs is organized by the size of the corresponding CGs. The IDs of the CGs that are of size i (i.e., the CGs in CGA[i]) are grouped together in an array, called the* size-i ID-array *of u.*

We assign the ID of a CG as follows. Let CGA[i][j] be the j-th entry in CGA[i]. We assign the CG stored in CGA[i][j] an *ID* j.

Constructing the CG-index, as shown in Algorithm 3, is straightforward and efficient. Let U be the CG in CGA[i][j]. We add the ID of U, which is j, to the end of the size-i ID-array of each node $u \in U$. The ID-arrays of the nodes are accessed via a link in the NA, while the nodes in the NA can be efficiently accessed via a hashtable.

After we obtain the CGs of the query nodes, we still need to find how the nodes in a CG are connected with each other. Let U be a CG and G_U be the subgraph that gives the *best* connection among the nodes in U. We call G_U the *correlation subgraph* (or *C-graph* for short) of U. Different methods may be used to compute G_U. For example, we may define G_U as the induced subgraph of $\mathcal{W}$ by U, or we may connect the correlated nodes in U by the shortest paths between them. Since we define correlation by relevance score based on RWR, we use an RWR-based approach to compute G_U by an algorithm similar to that in [1] (Section V.C).

Algorithm 3 BuildIndex

Input: $\mathcal{C}_G$.
Output: The CG-index built on $\mathcal{C}_G$.

1. Store $U \in \mathcal{C}_G$ in CGA[i], where $|U| = i$;
2. Sort the CGs in each CGA[i] in descending order of their correlation scores;
3. **for each** $j = 1, 2, \ldots,$ in each CGA[i] **do**
4. **for each** $u \in U$ in CGA[i][j] **do**
5. Insert u into the NA if u is not yet in the NA;
6. Add j to the end of the size-i ID-array of u;

However, after finding the groups, we also need to compute the connection between the groups. For this purpose, we model the relation between the CGs in a graph as follows.

Definition 5 (CG-Graph). *A* CG-graph *is a weighted, undirected graph* $\mathcal{G} = (V_\mathcal{G}, E_\mathcal{G})$ *defined as follows:*

- *The set of nodes* $V_\mathcal{G}$ *in* $\mathcal{G}$ *is the set of CGs* $\mathcal{C}_G$.
- *The set of edges* $E_\mathcal{G}$ *in* $\mathcal{G}$ *and the corresponding edge weight are defined as follows:*
  - $E_\mathcal{G} = \{(U, V) : U, V \in \mathcal{C}_G,$ *and* $((\exists (u, v) \in E_\mathcal{W}$ *s.t.* $u \in U$ *and* $v \in V)$ *or* $(\exists u \in U$ *s.t.* $u \in V))\}$.
  - *The weight of each edge* $(U, V) \in E_\mathcal{G}$ *is defined as* $w(U, V) = AVG\{(r(u, v) + r(v, u)) : u \in U, v \in V,$ *and* $((u, v) \in E_\mathcal{W}$ *or* $u = v)\}$.

Intuitively, in Definition 5 we construct the CG-graph $\mathcal{G}$ by connecting any two CGs if there is an edge between them in $\mathcal{W}$ or if they share any common nodes. In other words, two CGs are connected in $\mathcal{G}$ if and only if they are connected in the original graph $\mathcal{W}$. Since we measure the correlation by the relevance scores, we also assign the weights of the edges as the average relevance scores of the nodes that connect the two CGs.

4. QUERY PROCESSING

In this section, we discuss how we process a query using the CG-index and the CG-graph.

Given a set of query nodes, Q, we find the set of CGs that contain all nodes in Q by the algorithm *FindCG* as shown in Algorithm 4.

FindCG first finds the set of all maximal CNs in Q. The purpose of this step is to group the correlated nodes in Q first. Since Q is small, it is efficient to find all maximal CNs in Q.

The CNs that are maximal in Q may not be maximal in the global $\mathcal{W}$; therefore, in Lines 3-10 of Algorithm 4, FindCG uses the CG-index to find the CG that contains each of the maximal CNs in Q. For each maximal CN U, we begin with the size-$|U|$ ID-arrays of each node $u \in U$. We skip all size-i ID-arrays, for $i < |U|$, because the CGs whose IDs are in a size-i ID-array cannot contain U. The algorithm is simple, as we only need to intersect the size-i ID-arrays and obtain the first CG returned by the intersection. Finally, we return the set of all CGs obtained, which contains all nodes in Q.

According to the way that the CG-index is constructed, the first CG obtained by the intersection is the smallest CG

Algorithm 4 FindCG

Input: The CG-index, and a set of query nodes Q.
Output: The set of CGs that contains all nodes in Q, $\mathcal{C}_Q$.

1. $\mathcal{C}_Q \leftarrow \emptyset$;
2. Find the set of all maximal CNs in Q;
3. **for each** maximal CN, U, **do**
4. **for each** $i = |U|, |U| + 1, \ldots, n,$ where n is the size of the largest CG, **do**
5. Intersect the size-i ID-array of u, $\forall u \in U$;
6. **if**(the intersection returns an ID, j)
7. Add the CG in CGA[i][j] to $\mathcal{C}_Q$;
8. Go to *Line* 3 and continue with next U;
9. **else if**(the intersection reaches the end of the size-i ID-array of some u)
10. Go to *Line* 4 and continue with next i;
11. **return** $\mathcal{C}_Q$;

that contains U. Here, we favor a smaller CG because the smaller the CG, the larger proportion is U in the CG. When there is a tie between CGs of the same size, the first CG obtained has the highest correlation score since their IDs are ordered according to their correlation scores in the ID-arrays.

Next, we compute the best connection among the CGs. We model the problem as a Steiner tree problem as follows: given the CG-graph $\mathcal{G}$ and a set of CGs $\mathcal{C}_Q$, find a subtree of minimal weight which includes all CGs in $\mathcal{C}_Q$. However, before we can apply the Steiner tree model, we need to first transform the weight of each edge in $\mathcal{G}$ into its inverse, since we are finding the minimum-weight tree.

The minimum-weight Steiner tree problem has a fast algorithm [5] that achieves 2-approximation and a complexity of $|\mathcal{C}_Q|^2 \mathcal{O}(|V_\mathcal{G}| \log |V_\mathcal{G}|)$, where $|\mathcal{C}_Q|^2$ is usually a small constant.

Note that each node in the minimum-weight Steiner tree is a CG; for clearer visualization, we can display the query answer as the Steiner tree, but allow the user to click into a CG to view the detailed relationship between the nodes when he/she is interested.

5. EXPERIMENTAL RESULTS

We assess the performance of CG computation, index construction and query processing. We run all experiments on an AMD Opteron 248 with 1GB RAM, running Linux 64-bit.

We use the DBLP co-authorship dataset. The graph has approximately 316K nodes and 1,834K edges, where a node represents an author and the edge weight is the number of papers co-authored between two authors. Due to space limit, we only report part of our results.

5.1 Performance of Query Processing

In this experiment, we examine the performance of query processing. We select three sets of queries as follows.

- *CorQ*: for each query in the set, all the query nodes are randomly selected from some CG.
- *RelQ*: for each query in the set, the query nodes are randomly selected from some CG and those nodes that are related to the nodes in the CG.

- *RandQ*: for each query in the set, the query nodes are randomly selected from the set of nodes in the entire DBLP graph.

The three sets of queries represent three categories of correlation between the nodes in a query. For each query in *CorQ*, the nodes are highly correlated with each other within the query. For each query in *RelQ*, only part of the nodes in the query are highly correlated while the rest of the query nodes are only related to others (one-way relation). For each query in *RandQ*, the nodes within the query are likely not correlated nor related.

We further classify each set of queries into five groups (each containing 1000 queries) by varying the size of the queries from 2 nodes to 20 nodes. Figure 1 shows the results of the query response time averaged over 1000 queries. The response time increases linearly with the increase in the query size. However, the worst response time for queries of 20 nodes is still less than 700 μsec. Overall, the query performance is impressive as the response time is measured in μsec. This experiment demonstrates the efficiency of our CG-index and the C-graph extraction algorithm.

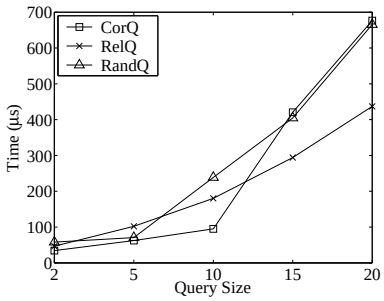

Figure 1: Response Time Vs Query Sizes

5.2 A Case Study

We describe a case study to assess the quality of the query answers. We select a query with the following four authors:

- Founders of *Google*: *Sergey Brin* and *Lawrence Page*
- Founders of *Yahoo!*: *Dave Filo* and *Jerry Chih-Yuan Yang*

Due to space limit, we only show one full C-graph, at the top shaded region of Figure 2, which corresponds to the CG, {*Sergey Brin*, *Lawrence Page*}. The correlation score of the C-graph is 0.01. The C-graph consists of nine persons. Among the nine persons, *Page* is related to *Brin*, *Cho*, *Garcia-Molina*, *Motwani*, and *Winograd*. However, apart from *Brin*, *Page* has only one collaboration with the others. The reason that *Page* is related to them is because *Page* has only three publications and these five persons are all the co-authors. *Page* has two out of three papers with *Brin*, thus they are correlated. But the other scholars are not related to *Page* or *Brin* since they have a large number of publications with other people. As a result, although the C-graph may seem to be complicated, it reflects the true and significant information of *Brin* and *Page* regarding to their academic publications.

For the founders of *Yahoo!*, since *Filo* and *Yang* do not have any paper co-authored, they are separated in two C-graphs. We show the two C-graphs by the shaded region at the bottom of Figure 2 (only those very closely related authors are shown), which shows that both *Filo* and *Yang* are related to *De Micheli* who links the two C-graphs together.

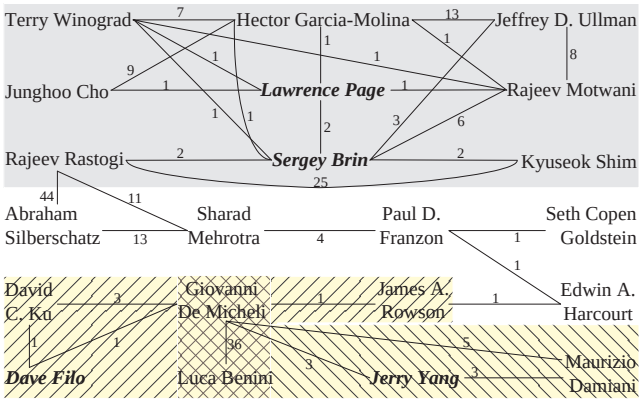

Figure 2: Relationship between the Founders of *Google* and *Yahoo!*

The overall answer graph depicted in Figure 2 reveals the relationship between the founders of *Google* and *Yahoo!*. Every edge in Figure 2 has a relevance score of at least σ, which implies a strong correlation or relation between the two authors linked by the edge. The computation of this answer graph takes only 0.016 second.

6. CONCLUSIONS

We propose an effective group formation technique that classifies the nodes in a large graph into meaningful groups called CGs. The notion of CG reflects the correlation among all the nodes within a CG. We develop an index for efficient query processing. Experimental results verify the meaningfulness of the CGs and C-graphs and demonstrate the efficiency of our algorithms.

7. REFERENCES

[1] J. Cheng, Y. Ke, W. Ng, and J. X. Yu. Context-aware object connection discovery in large graphs. In *ICDE*, pages 856–867, 2009.

[2] C. Faloutsos, K. S. McCurley, and A. Tomkins. Fast discovery of connection subgraphs. In *KDD*, pages 118–127, 2004.

[3] J. M. Kleinberg, R. Kumar, P. Raghavan, S. Rajagopalan, and A. Tomkins. The web as a graph: Measurements, models, and methods. In *COCOON*, pages 1–17, 1999.

[4] Y. Koren, S. C. North, and C. Volinsky. Measuring and extracting proximity in networks. In *KDD*, pages 245–255, 2006.

[5] K. Mehlhorn. A faster approximation algorithm for the steiner problem in graphs. *Inf. Process. Lett.*, 27(3):125–128, 1988.

[6] M. E. J. Newman. Fast algorithm for detecting community structure in networks. *Physical Review E*, 69:066133, 2004.

[7] D. J. D. S. Price. Networks of scientific papers: The pattern of bibliographic reference indicates the nature of the scientific research front. *Science*, 149.

[8] H. Tong and C. Faloutsos. Center-piece subgraphs: problem definition and fast solutions. In *KDD*, pages 404–413, 2006.

[9] H. Tong, C. Faloutsos, and J.-Y. Pan. Fast random walk with restart and its applications. In *ICDM*, pages 613–622, 2006.

[10] S. Wasserman and K. Faust. Social network analysis. *Cambridge University Press.*