Community Search for Large Profiled Graphs

ABSTRACT

abstract

Proceedings of the VLDB Endowment, Vol. 9, No. 12

1. THE PCQ PROBLEM

DEFINITION 1 (k-CORE [5, 1]). *Given an integer k ($k \geq 0$), the k-core of G, denoted by H_k, is the largest subgraph of G, such that $\forall v \in H_k$, $deg_{H_k}(v) \geq k$.*

DEFINITION 2 (PROFILE-TREE). *xxxx*

DEFINITION 3 (INDUCED ROOTED SUBTREE). *Given two P-trees T and S rooted at r_t and r_s. T is the induced rooted subtree of S iff* [1] *there exist a one-to-one mapping φ: $V_s \rightarrow V_t$ and l denotes the label of tree node x, three constraints hold:*

- $\varphi(r_s) = r_t$;
- $\forall x \in V_s$, $l(x) = l(\varphi(x))$;
- $\forall (x, y) \in E_s$, $(\varphi(x), \varphi(y)) \in E_t$.

Inuduced rooted subtree preserves the parent-child relationships as well as corresponding labels. In addition, the root node of the trees must be preserved. Unless otherwise specified, all uses of the term "subtree" refer to "induced rooted subtree". P-trees T and S are *isomorphic* iff T is the subtree of S and S is the subtree of T simultaneously.

DEFINITION 4 (MAXIMUM COMMON SUBTREE). *Given a graph G, D is a P-tree database which pairs with vertices in G. T is the common subtree of D such that $\forall t \in D$, T is the subtree of t. Furthermore, T is the maximum common subtree of D (denoted as $\Gamma(G)$) if there exists no other common subtree T' such that T is the subtree of T'.*

PROBLEM 1 (PCQ). *Given a profiled graph $G(V, E)$, a profile tree T, a positive integer k, and one query node $q \in G$, find a set $\mathcal{G}$ of graphs, such that $\forall G_q \in \mathcal{G}$, the following properties hold:*

- ***Connectivity cohesiveness**. (1)$G_q \subseteq G$ is connected and contains q, (2)$\forall v \in G_q$, $deg_{G_q}(v) \geq k$;*
- ***Maximal structure**. There exists no other $\tilde{G}_q$ satisfying **connectivity cohesiveness** such that $G_q \subset \tilde{G}_q$ if $\Gamma(G_q)$ and $\Gamma(\tilde{G}_q)$ are isomorphic;*
- ***Semantic cohesiveness**. There exists no other $\tilde{G}_q$ satisfying **connectivity cohesiveness**, such that $\Gamma(G_q)$ is the subtree of $\Gamma(\tilde{G}_q)$.*

[1] if and only if.

2. BASIC ALGORITHM

Algorithm 1 Basic query algorithm

```
1: function QUERY(G, q, k)
2:     G ← ∅
3:     init subtree T_q using q's P-tree root;
4:     Find k − core C_q containing q from G;
5:     MINE(C_q, q, T_q, G, k);
6:     return G;
7: function MINE(G, q, T, G, k)
8:     if |G| ≤ k then return;
9:     init Γ ← generate new subtrees from T;
10:    if Γ = ∅ then
11:        G ← G;
12:    else
13:        flag ← true;
14:        for each T' in Γ do
15:            find k − core C'_q containg q from G;
16:            if C'_q ≠ ∅ then
17:                flag ← false;
18:                MINE(G', q, T', G, k)
19:        if flag = true then
20:            G ← G' iff T' is a new maximal subtree;
```

3. BASIC INDEX

Algorithm 2 index construction: basic

```
1: function BUILDINDEX(G)
2:     compute the core number of vertices in G;
3:     for i = 0 to k_max do
4:         init IT_i ← ∅;
5:         compute all i − cores;
6:         for each each i − core, G_i do
7:             for each vertex u in G_i do
8:                 INSERT(u, IT_i);
9:     delete empty index node in IT_i;
10:    return all IT;
11: function INSERT(u, IT)
12:    for each P-tree item p in DFS path of u's P-tree do
13:        if IT does not have node p then
14:            create an index node it_p with p;
15:            insert it_p in IT;
16:        if p is a leaf node of u's P-tree then
17:            locate the index node It_p;
18:            add u in it_p.vertexSet;
```

Complexity analysis. Space complexity is $O(k_{max} \cdot l)$ where l is size of the minimum supertree of all P-trees.

Time complexity for basic algorithm is $O(m(k_{max} + \hat{l} + 1))$ where $\hat{l}$ is the average number of P-trees.

4. ADVANCED INDEX

Algorithm 3 index construction: advanced

```
1: function BUILDINDEX(G)
2:     compute core number of all vertices in G;
3:     k ← 0;
4:     for each v ∈ G do MAKESETv
5:     use union-find structure;
6:     for for each v do
7:         create core[v] index nodes it_0, it_1, it_2, ··· , it_core[v];
8:     CONSTRUCTTREE()
9: function CONSTRUCTTREE()
```

Complexity analysis. Space complexity is $O(k_{max} \cdot l)$ as well.

Time complexity for advanced algorithm is $O(m \cdot a(n) + k_{max} \cdot l)$ where l is the minimum suptree of all p-trees.

5. HARDNESS OF THE PROBLEM

5.1 Preliminaries

DEFINITION 5 (#P [6]). *A counting problem that can be computed by nondeterministic Turing machine running in polynomial time is categorised in the class #P.*

DEFINITION 6 (#P-HARD [6]). *A problem is #P-hard if all problems in #P reduce to it.*

DEFINITION 7 (#P-COMPLETE [6]). *A problem is #P-complete if all problems in #P reduce to it and it belongs to #P.*

In computational complexity theory and computability theory, a counting problem is a problem only returns the number of all solutions and it is $\#P$. Valiant further defined the class #P-complete as the "hardest" problem in #P as the concept of NP-complete is introduced in NP problems. Garey et al. [2] and Papadimitriou et al. [3] proved that if a counting problem is #P-complete, then its associated problem of mining all solutions must be NP-hard. Based on above conclusion, GuiZhen Yang has proved that mining frequent itemsets including subtrees is NP-hard [7]. As for our problem which holds three property defined in Problem 1, all validated communities are required to computed. Thus we follow the same principle that if we can prove that (in worst case) counting the number of all required communities is #P-complete, then our problem is NP-hard.

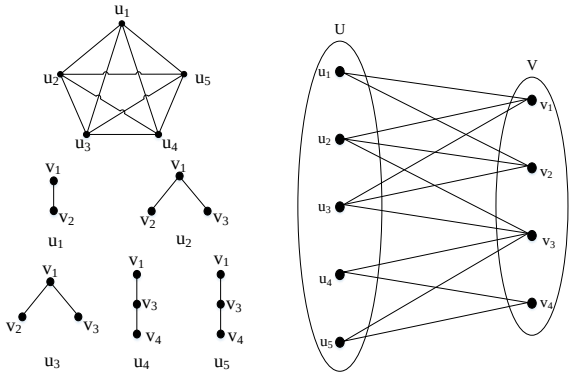
Bipartite graph. A bipartite graph can be denoted as a triple, $G = (U, V, E)$, where vertices can be partitioned to two disjoint sets U and V, and E is the set of edges between vertices in U and V, i.e., $E \subseteq U \times V$.

A bipartite clique is a subgraph of a bipartite graph such that every vertices in two distinct vertex sets are adjacent. Furthermore, a bipartite clique $G' = (U', V', E')$ is a maximal bipartite clique in a given bipartite graph, if there exists no other bipartite clique $G'' = (U'', V'', E'')$ such that $U' \subseteq U''$, $V' \subseteq V''$, $E' \subseteq E''$ simultaneously.

Construction of bipartite graph. Since each node in P-tree is unique and has fixed location in P-tree, knowing all p-tree nodes is enough to reconstruct the original P-tree. Then we can simply construct a bipartite graph $G = (U, V, E)$ from the profiled graph where U is the set of all users in the profiled graph and V is the set of all unique P-tree nodes. Edges in E represent that users in U own the P-tree nodes in V. The construction process can be done in linear time. Note that this constructed bipartite graph is not equivalent to the associated profiled graph, because connectiveity of vertices in the profiled graph is not presented in $G = (U, V, E)$. But in the case that the profiled graph is a complete graph which means each induced subgraph is a clique, the profiled graph can be constructed to a bipartite graph without losing generality. We illustrute it in Example 1.

EXAMPLE 1. *The profiled graph and corresponding P-tree forests are shown in Figure 1(a). Since original graph is a complete graph, every subgraph is connected. Thus the constructed bipartite graph shown in Figure 1(b) is exactly equivalent to the orginal profiled graph.*

Based on the assumption and this one-to-one correspondence, we can reduce the problem of computing the number of all maximal bipartite cliques containing query node q in the bipartite graph to the problem of computing the number of required communities with q in the profiled graph. The latter problem will be proved #P-complete.



(a) Profiled graph (b) Bipartite graph

Figure 1: Construction of the bipartite graph

5.2 Proof

THEOREM 1 ([4]). *The problem of counting the number of maixmal bipartite cliques in a given bipartite graph is #P-complete.*

Let $C_i(G)$ denote a set of all qualified communities and $\forall C \in C_i(G)$, $|C| = i$; $M_i(G)$ represents a set of all qualified communities and $\forall M \in M_i(G)$, $|M| \geq i$. Clearly, $M_k(G)$ is the set of all targeted communities which hold the problem definitions. Based on Theorem 1, we have Corollary 1.

COROLLARY 1. *It is #P-complete to counting the* $\sum_{i=1}^{|G|} |C_i(G)|$.

Now we construct a new bipartite graph followed the strategy in [7] and represent it as a binary martix. Since the required community contains q, the maxinum common subtree of the community must be the subtree of q's. To make sure that the returned communities includes q. Let $V = \{v_1, v_2, ..., v_n\}$ be the set of q's P-tree nodes and $U = \{u_1, u_2, ..., u_m\}$ be all vertices in the profiled graph. D denotes the database which only recording the relationship between U and corresponding tree nodes V. Note that the construcution of D can be done from the original profiled graph in polynomial time. Then, we generate m new items namely $L = \{l_1, l_2, ..., l_m\}$ and m new vertices $S = \{s_1, s_2, ..., s_m\}$. Let function w denote expanded items of the vertex in U^+. As shown in Figures 2, the new bipartite graph D^+ is constructed as follows:

(1). $U^+ = U \cup S$.

(2). For $i \in [1, m], W(u_i) = D_i \cup L, W(s_i) = V \cup (L - l_i)$.

	v_1	v_2	$\cdots$	v_n	l_1	l_2	$\cdots$	l_m
u_1					1	1	$\cdots$	1
u_2			D		1	1	$\cdots$	1
$\vdots$					$\vdots$	$\vdots$	$\cdots$	$\vdots$
u_m					1	1	$\cdots$	1
s_1	1	1	$\cdots$	1	0	1	$\cdots$	1
s_2	1	1	$\cdots$	1	1	0	$\cdots$	1
$\vdots$	$\vdots$	$\vdots$	$\cdots$	$\vdots$			$\ddots$	
s_m	1	1	$\cdots$	1	1	1	$\cdots$	0

Figure 2: Binary matrix representation of bipartite graph D^+.

LEMMA 1. *$\forall a \in [0, m-\sigma]$, if a commmunity in $C_{\sigma+a}(G)$ whose maximum common subtree is I, $\sigma \in [1, m]$, then for any arbitrary itemset $L_a \subseteq L$, $|L_a| = a$, there must exist an communitiy in $C_{\sigma+m}(D^+)$ such that $I \cup L_a$ is its maximum common subtree.*

PROOF. If I is the maximum common subtree of a community in $C_{\sigma+a}(G)$, which means I is shared by $\sigma + a$ vertices ranging from u_1 to u_m. Then $I \cup L_a$ is still shared by these vertices because $L_a \subseteq L$. As for vertices from s_1 to s_m, I is shared by all of them. Note that $W(s_i) = V \cup (L - l_i)$ and $|L_a| = a$, thus there are $m - a$ vertices from s_1 to s_m that do not contains all items in L_a. In conclusion, for any any arbitrary itemset L_a, $I \cup L_a$ is shared by $\sigma + m$ vertices in D^+ which means these vertices group a community whose maximal common subtree is $I \cup L_a$. □

LEMMA 2. $M_{\sigma+m}(D^+) = C_{\sigma+m}(D^+)$

Lemma 2 implies that if there is a community in $C_{\sigma+m}(D^+)$ whose maximum common subtree is I' iff if there exists a community in $M_{\sigma+m}(D^+)$ whose maximum common subtree is I' as well.

PROOF. We prove necessity of Lemma 2 first and sufficiency follows.

• **Necessity.** It is obvious from the definition.

• **Sufficiency.** I' can be constructed as $I' = I \cup L_a$, where L_a is an arbitrary itemset and $|L_a| = a$. Since I' is shared by at least $\sigma + m$ vertices in D^+ and I' is shared by $m - a$ vertices from s_1 to s_m, I' is shared by at least $\sigma + a$ vertices from u_1 to u_m which group a community in D as well. From Lemma 1, if I' is shared by $\sigma+a'(a' > a)$ vertices in D, I' will be shared by $\sigma+m$ vertices in D^+ which means there exists a community in $C_{\sigma+m}(D^+)$ sharing I'. That proves the necessity part. □

Based on Lemma 2, we have the following equation.

LEMMA 3. *Let D denote the original database and D^+ be the expanded database, $|U| = m$, for $\sigma \in [1, m]$:*

$$|M_{\sigma+m}(D^+)| = \sum_{a=0}^{m-\sigma} |C_{\sigma+a}(D)| \cdot \binom{m}{a}.$$

THEOREM 2. *$G = U, V$ denotes the profiled graph, $|U| = m$, the problem of computing $|M_k(G)|$ where $k \in [1, m]$, is $\#P$-complete.*

PROOF. Checking whether a community is satisfying all properties or not can be done in polynomial time. Therefore, the problem of computing $|M_k(G)|$ is in #P. We now reduce the problem of counting the number of $\sum_{i=1}^{|G|} |C_i(G)|$ as stated in Corollary 1 to the problem of the counting of computing $|M_k(G)|$, thereby proving that the latter one is #P-hard.

First, we can construct D from the orginal profiled graph G in polynomial time. Let $C_\sigma = C_\sigma(D)$, $M_\sigma = M_{\sigma+m}(D^+), \sigma \in [1, m]$. Based on Lemma 3, we have following linear eqtions.

$$\begin{bmatrix} C_1 \\ C_2 \\ C_3 \\ \vdots \\ C_m \end{bmatrix} = \begin{bmatrix} \binom{m}{0} & \binom{m}{1} & \binom{m}{2} & \cdots & \binom{m}{m-1} \\ 0 & \binom{m}{0} & \binom{m}{1} & \cdots & \binom{m}{m-2} \\ 0 & 0 & \binom{m}{0} & \cdots & \binom{m}{m-3} \\ & & \ddots & & \\ 0 & 0 & 0 & \cdots & \binom{m}{m-(m-1)} \end{bmatrix}^{-1} \cdot \begin{bmatrix} M_1 \\ M_2 \\ M_3 \\ \vdots \\ M_m \end{bmatrix}$$

If we have a polynomial-time algorithm for computing $|M_k(G)|$, we can compute $(M_1, M_2, \cdots, M_m)$ in polynomial time. Since $\forall x \in [0, m-1]$, $\binom{m}{x}$ can be compute in O(m), $(C_1, C_2, \cdots, C_m)$ and $\sum_{\sigma=1}^{m} C_\sigma$ can be computed in polynomial time and therefore the problem of computing the number of $|M_k(G)|$ is #P-complete. □

In conclusion, since the problem of counting the number of all qualified communities is #P-complete, the problem of mining all PCQ communities is NP-hard.

6. REFERENCES

[1] V. Batagelj and M. Zaversnik. An o(m) algorithm for cores decomposition of networks. *arXiv*, 2003.

[2] M. R. Garey and D. S. Johnson. A guide to the theory of np-completeness. *WH Freemann, New York*, 70, 1979.

[3] C. H. Papadimitriou. *Computational complexity*. John Wiley and Sons Ltd., 2003.

[4] J. S. Provan and M. O. Ball. The complexity of counting cuts and of computing the probability that a graph is connected. *SIAM Journal on Computing*, 12(4):777–788, 1983.

[5] S. B. Seidman. Network structure and minimum degree. *Social networks*, 5(3):269–287, 1983.

[6] L. G. Valiant. The complexity of computing the permanent. *Theoretical computer science*, 8(2):189–201, 1979.

[7] G. Yang. The complexity of mining maximal frequent itemsets and maximal frequent patterns. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 344–353. ACM, 2004.