人工知能学会研究会資料
SIG-FPAI-B403-10

疑似クリーク制約を用いたクリーク族の全列挙

Maximal Clique Sets with Pseudo Clique Constraints

ジェイ 泓杰[1*] 原口 誠[1] 大久保 好章[1] 富田 悦次[2]
Hongjie Zhai[1] Makoto Haraguchi[1] Yoshiaki Okubo[1] Etsuji Tomita[2]

[1] 北海道大学大学院情報科学研究科
[1] Graduate School of Information Science and Technology, Hokkaido University
[2] 電気通信大学先進アルゴリズム研究ステーション
[2] The Advanced Algorithms Research Laboratory,The University of Electro-Communications

Abstract: Many variants of pseudo-cliques have been introduced as a relaxation model of cliques to detect communities in real world networks. For most types of pseudo-cliques, enumeration algorithms can be designed just similar to maximal clique enumerator. However, the problem of enumerating pseudo-cliques is computational hard, because the number of maximal pseudo-cliques-cliques is huge in general. Furthermore, because of the weak requirement of k-plex, sparse communities are also allowed depending on the parameter k. To obtain a class of more dense pseudo cliques and to improve the computational performance, we introdue a derived graph whose vertices are cliques in the original input graph. Then our target pseudo must be a clique or a pseudo clique of the derived graph under an additional constraint requiring density in the original graph. An enumerator for this new class is designed and its computational efficientcy is experimentally verfied.

1 Introduction

For graph theory, cliques are subgraphs in which all vertices connect to each other. It is usually used to detect densely connected communities. However, for real-world datasets, there exists many noises to make many dense community not to be a clique. The strict definition of clique is not suited for real applications. To address this issue, many relaxion models of clique, alse called pseudo-cliques, are introduced [3]. Every pseudo-clique model weaken some requirement of clique. k-clique and k-clan are defined by weakening reachability. k-core, k-plex are defined by relaxing the requirement of closeness. In cases of k-cores and k-plexes, some vertices may not be connected. In this paper, we will mainly discuss k-plex.

K-plex was introduced by Seidman [1]. K-plex is a subgraph in which each member is connected to at least $n - k$ other members. When $k = 1$, k-plex becomes clique. For k-plex model with small k values as (k=2,3), k-plexes can detect the densely connected subgraphs. However, when k grows larger, sparse graph such as chain or circle will become k-plexes. The number of k-plexes also grows exponentially. Even though clique enumeration can be done efficiently, the task of finding all the maximal k-plexes for larger k is actually impractical, because k-plex allows much more combination of vertices than clique.

In this paper, to obtain densely connected k-plexes, we consider the k-plexes are only combined from maximal cliques whose size is larger than a given lower bound. In an other words, our target is to detect maximal clique sets under the k-plex constraint. By introducing the concept of clique-k-plex and k-clique-graph, the maximal clique sets can be found efficiently with a simple algorithm. Moreover, to exclude the k-plexes which have sparsely connected core, we also introduce bond measure as an extra constraint into our algorithm. The bond constraint can cut off most combination of sparsely connected k-plexes with small "cores". The rest of this paper is organized as follows. Basic definitions, notations are presented in Section 2.1. Section 2.2 presents the basic idea of clique-k-plex and k-clique-graph. Our algorithms are given in Section 2.3 and an additional constraints with bond measure is introduced in Section 2.4. We also

*連絡先：北海道大学大学院情報科学研究科
〒 060-0814 札幌市北区北１４条西９丁目
E-mail: zhaihj@kb.ist.hokudai.ac.jp

show experiment of our algorithms in Section 3. Finally the paper is concluded with a summary and directions for future works.

2 Proposed Method

2.1 Notations

Let $G = (V, E)$ be a simple undirected graph. Vertices $V = \{v_1, v_2, \cdots, v_n\}$ and edges $E = \{e_1, e_2, \cdots, e_n\}$. $|G|$ denotes the number of vertices in G. A subset of vertices $c \subseteq V$ is a clique if all the pairs vertices of its are connected to each other. A clique is maximal if it is not a subset of another clique. $C(G) = \{c_1, c_2, \cdots, c_j\}$ denotes all the maximal cilques of G, and all its members are sorted by size $|c_1| \geq |c_2| \geq \cdots \geq |c_j|$. The problem of enumerating all maximal cliques is "Maximal Clique Problem" and the base algorithm is call BK which was firstly introduced in 1973 [2].

Definition 1. *A subset of vertices $S \in V$ is a k-plex if $deg_{G[s]}(v) \geq |s| - k$ for every vertex v in S.*

A k-plex is called maximal if it is not a subgraph of other k-plex. Because of the weak requirement of k-plexes, target vertices is not guranteed to be connected. k-plex is anti-monotonic, it means that if vertices set X is k-plex, for any subset $Y \subset X$, $\forall x \in X - Y$, $x \cup Y$ is also a k-plex.

Definition 2. *A subset of cliques $T \in C$ is a clique-k-plex if vertices set $S = t_1 \cup t_2 \cup \cdots \cup t_j$ is a k-plex in G.*

A clique-k-plex is maximal if it is not a subgraph of any other clique-k-plexes. But it is clear that maximal clique-k-plexes are not always maximal k-plexes in graph G. Different maximal clique-k plexes may represent same vertices set in G. Eventhough deleting duplicated point can be easily done as post-process, here we still treat them as different clique-k-plex. According to the definition, clique-k-plex is also anti-monotonic. From now on, we will say a clique set C is k-plex in G by meaning $c_1 \cup c_2 \cup \cdots \cup c_j$ is a k-plex in G.

2.2 Basic Idea

For a given graph G, k and a size parameter L, we enumerate all the maximal cliques c_i in G with constraint $|c_i| \geq L$ and search for all the maximal clique-k-plexes from this clique set. If $L = 1$, we are using all the maximal cliques. Size constraints on cliques is extremely useful for the graphs that contain large amount of triangles or solitude vertices. We should also point out that enumerating all maximal clique-k-plexes equals the process that searching for all maximal k-plexes on a graph in which all the vertices is cliques. To give a clear state, we have the following definition:

Definition 3. *A graph $C_G = (C_V, C_E)$ is called k-clique-graph if every vertex $c_i \in C_V$ is a clique in graph G and a edge exists between two vertices c_i and c_i if and only if $c_i \cup c_j$ is a clique-k-plex.*

According to Definition 3, only the pairs that can be k-plexes is connected to each other. From anti-monotone of clique-k-plex, we know that for a vertices set $C \subseteq C_V$, C is a clique on k-clique-graph C_G if C is a k-plex in G. When searching for maximal clique-k-plexes, this property allows us to limit candidates in a small range, which is all the maximal clique on k-clique-graph. We call the cliques on k-clique-graph "meta-clique" because they are the cliques of cliques for graph G. We have to point out that cliques on k-clique-graph are not always clique-k-plexes when the size of meta-cliques is larger than 2. We should validate all the meta-cliques if they are clique-k-plexes or not.

2.3 Algorithms

Algorithm 1 gives a general steps for enumerating all maximal clique-k-plexes. In this algorithm, Build K-Clique-Graph (Algorithm 2) will build a k-clique-graph from normal graph. Then it is able to list all meta-cliques by Enumerate Maximal Clique-K-plexes (Algorithm 3). In Algorithm 2, we firste enumerate all the maximal cliques, and then validate all the cliques pair-wisely if they can be a k-clique-graph or not. In Algorithm 3, we search for all maximal "meta-cliques" on given graphs. When maximal meta-cliques are clique-k-plexes, it can be directly returned as result. When it is not a clique-k-plexes, we need to search for all maximal clique-k-plexes in this meta-clique. In most time, the meta-clique only contains a small set of C_V and can be enumerated efficiently. Here algorithm *EnumerateAllCliqueKPlexes* is a bk-like algorithm for detecting all maximal clique-k-kplexes. The pseudo code of this algorithm is shown in Algorithm 4.

In all these algorithms, FindMaximalCliqeus can be any algorithm to enumerate all maximal cliques on undirected graph. Here we use the general BK algorithm which was introduced in [2].

Algorithm 1: MaximalCliqueKPlex

Input: A undirected graph $G = (V, E)$
Output: All maximal clique-k-plexes
C_G = BuildKCliqueGraph (G);
result = EnumerateMaximalCliqueKplexes (C_G);
return *result*;

Algorithm 2: BuildKCliqueGraph

Input: Graph G, Integer K, L
Output: k-clique-graph
Initialize $C_V = \emptyset, C_E = \emptyset$;
cand = FindMaximalCliques (G);
while $cand \neq \emptyset$ **do**
  curCand = a clique in cand;
  $cand = cand - curCand$;
  if $|C_V| < L$ **then**
    continue;
  end
  for *all cliques* c_i *in* C_V **do**
    if $c_i \cup curCand$ *is k-plex* **then**
      $C_E = C_E + e(i, \|C_V\|)$;
    end
  end
  $C_V = C_V$ + curCand;
end
return $G(C_V, C_E)$;

Algorithm 3: EnumerateMaximalCliqueKPlexes

Input: k-clique-graph C_G
Output: Set of maximal clique-k-plexes
Initialize $result = \emptyset$;
cand = FindMaximalCliques (C_G);
while $cand \neq \emptyset$ **do**
  curCand = a "meta-clique" in cand;
  if $|curCand| \leq 2$ **then**
    $result = result + curCand$;
  end
  else
    if *curCand is clique-k-plex* **then**
      $result = result + curCand$;
    end
    else
      $result = result +$
      $EnumerateAllCliqueKPlexes(curCand)$;
    end
  end
  $cand = cand - curCand$;
end
return *result;*

2.4 Bond Constraint for Buiding k-Clique-Graph

In many cases, our target is to find dense part of graph. However, as we have already discussed, k-plex does not have constraint on connectivity. In the algorithm 2 (Build K Clique Graph), cliques will become k-clique-graph if they can be clique-k-plexes regardless of if they are connected or not. For this reason, we use bond [5], an extended Jaccard coefficient, to calculate the degree of overlappingness of two vertices sets. The definition of Bond is shown in 4. By only considering the clique pairs whose bonds are above certain degree, it allows us to focus on the dense connected part of graphs. Algorithm 5 shows the algorithm for building k-clique-graph with bond measure.

Definition 4. *For two vertices set A and B, the Bond Measure is defined as* $Bond(A, B) = \frac{|A \cap B|}{|A \cup B|}$

For dense graph, there usually exists large amount of maximal cliques. In algorithm 2, all cliques are checked pair-wisely if they are clque-k-plexes or not. This will take a long time. By introducing bond measure, we are able to cut most useless candidates to improve the performance.

3 Experiment

In order to evaluate the efficiency and scalability of our algorithm on several benchmark graphs. The basic statistics of graph datasets is shown in Table 1. Test instances mainly contains 3 parts. The first class comes from DIMACS [6]. The DIMACS graphs are considered as the standard test cases for clique and pseudo-clique algorithms. Second class is real-life social network. Erdos [4] is a co-authorship network, in which Paul Erdős is a prolific mathematician lying at the core of the entire network.

Algorithm 4: EnumerateAllCliqueKPlexes

Input: k-clique-graph $C_G(C_V, C_E)$
Output: Set of maximal clique-k-plexes
Initialize $cand = C_V$, $result = \emptyset$, $comp = \emptyset$, $not = \emptyset$;
Call BKEnumerate (comp, cand, not, result);
return *result*;
Procedure *BKEnumerate (comp, cand, not, result)*
  if $cand = \emptyset$ **then**
    if $not = \emptyset$ **then**
      result = result + comp;
      return;
    end
  end
  else
    while $cand \neq \emptyset$ **do**
      $v = avertexincand$;
      $newcomp = comp + v$;
      $cand = cand - v$;
      newcand = all vertices in cand that can be clique-k-plexes with *comp*;
      newnot = all vertices in not that can be clique-k-plexes with *comp*;
      BKEnumerate (newcomp, newcand, newnot, result);
      $not = not + v$;
      cand = all vertices in cand that can not be clique-k-plexes with *comp*;
    end
  end
  return;

Algorithm 5: BuildKCliqueGraphWithBond

Input: Graph G, Integer K, L, Float B
Output: k-clique-graph
Initialize $C_V = \emptyset, C_E = \emptyset$;
cand = FindMaximalCliques (G);
while $cand \neq \emptyset$ **do**
  curCand = a clique in cand;
  $cand = cand - curCand$;
  if $|C_V| < L$ **then**
    continue;
  end
  for *all cliques* c_i *in* C_V **do**
    if $Bond(c_i, curCand) > B$ **then**
      if $c_i \cup curCand$ *is k-plex* **then**
        $C_E = C_E + e(i, \|C_V\|)$;
      end
    end
  end
  $C_V = C_V$ + curCand;
end
return $G(C_V, C_E)$;

The whole program is impremented in C with libGC and compiled with Clang. Our experiments are performed on Linux with single core 2.4GHz CPU and 8 Gbytes memory. The program is termined if it runs for more than one hour. The number of vertices and egdes of graphs and run time are shown in Table 3 and Table 4. As a comparation, the runtime of basic algorithm [7] are also presented in Table 2. We show more details for Erdos971. In Table 5 and Table 6, we show the runtime for $k = \{2, 3, 4, 5\}$ and $L = \{1, 2, 3, 4\}$ under different bond constraint. The similar results for number of patterns are also shown in Table 7 and Table 8.

dataset	# of V	# of E	deg_{avg}	deg_{max}
hamming6-2	64	1824	57	57
hamming6-4	64	704	22	22
ERDOS971	472	1314	5.59	41
ERDOS981	485	1381	5.71	42
ERDOS991	492	1417	5.76	42

Table 1: Statistics of the graph datasets

dataset	$K=2$	$K=3$	$K=4$	$K=5$
hamming6-2	506	≥ 3600	≥ 3600	≥ 3600
hamming6-4	10	1	9	20
ERDOS971	10	1122	≥ 3600	≥ 3600
ERDOS981	9	1259	≥ 3600	≥ 3600
ERDOS991	10	1323	≥ 3600	≥ 3600

Table 2: Runtime of basic algorithm (For comparation)

dataset	$k=2$		$k=3$		$k=5$	
L	1	4	1	4	1	4
hamming6-2	N^*	N	N	N	N	N
hamming6-4	0.5	0.1	2.6	2.2	4.8	2.0
ERDOS971	3.7	0.6	9.6	0.8	2711	1048
ERDOS981	4.1	0.7	10	0.9	2736	746
ERDOS991	4.5	0.7	11	1.0	3208	846

Table 3: Runtime (seconds) on graph datasets, $B = 0$

$*$: N means the algorithm fails to return results in 3600 seconds (1 hour).

dataset	$k=2$		$k=3$		$k=5$	
L	1	4	1	4	1	4
hamming6-2	N	N	N	N	N	N
hamming6-4	0.02	0.01	2.6	2.2	4.7	2.0
ERDOS971	1.1	0.3	1.6	0.5	16	9.6
ERDOS981	1.1	0.4	1.9	0.6	18	11
ERDOS991	1.2	0.4	2.1	0.6	18	11

Table 4: Runtime (seconds) on graph datasets, $B = 0.1$

Size Constraint	k=2	k=3	k=4	k=5
L=1	3.7	9.6	180	2711
L=2	3.4	8.2	176	2714
L=3	1.8	2.2	113	2424
L=4	0.6	0.8	11	1048

Table 5: Runtime (seconds) for Erdos971, $B = 0$

Size Constraint	k=2	k=3	k=4	k=5
L=1	1.1	1.6	10	16
L=2	1.0	1.5	10	16
L=3	0.6	1.0	9.2	14
L=4	0.3	0.5	5.9	9.6

Table 6: Runtime (seconds) for Erdos971, $B = 0.1$

Size Constraint	k=2	k=3	k=4	k=5
L=1	721	6050	284173	2297910
L=2	683	5913	284136	2297873
L=3	376	2539	243957	2199459
L=4	175	1095	35389	1198043

Table 7: Number of patterns for Erdos971, $B = 0$

Size Constraint	k=2	k=3	k=4	k=5
L=1	708	2735	26628	26155
L=2	672	2699	26592	26018
L=3	396	2273	26165	25753
L=4	175	1116	19172	19333

Table 8: Number of patterns for Erdos971, $B = 0.1$

4 Conclusion and Future Work

By restricting target on the combination of cliques, pseudo-cliques can be found efficiently for large graph. We introduce the concept of clique-k-plex and k-clique-graph, and propose an maximal clique-k-plexes enumeration algorithm. We also introduce constraint on size of clique and bond measure to analysis large network. The future work mainly focuses on improving performance and implement the BK-like k-plex enumerating algorithm under k-clique-graph framework.

References

[1] Seidman, Stephen B., and Brian L. Foster. A graph - theoretic generalization of the clique concept. *Journal of Mathematical sociology* 6.1 (1978): 139-154.

[2] Bron, Coen, and Joep Kerbosch. Algorithm 457: finding all cliques of an undirected graph. *Communications of the ACM* 16.9 (1973): 575-577.

[3] J. Pattillo, N. Youssef, and S. Butenko. Clique relaxations in social network analysis. In M. T. Thai and P. M. Pardalos (Eds.), *Handbook of Optimization in Complex Networks: Communication and Social Networks*, pages 143-162. Spinger Science + Business Media, 2012.

[4] Vladimir Batagelj and Andrej Mrvar. Pajek datasets. *http://vlado.fmf.uni-lj.si/pub/networks/data/* (2006)

[5] Omiecinski, Edward R. Alternative interest measures for mining associations in databases. *Knowledge and Data Engineering*, IEEE Transactions on 15.1 (2003): 57-69.

[6] Cliques, coloring, and satisfiability: Second DIMACS implementation challenge, 1995.

[7] McClosky, Benjamin, and Illya V. Hicks. Combinatorial algorithms for the maximum k-plex problem. *Journal of combinatorial optimization* 23.1 (2012): 29-49.