Heuristic Algorithm for Extracting a Subset of Maximal Cliques Inside Graphs

Trinh Anh Phuc*, Dinh Viet Sang
Hanoi University of Science and Technology, No. 1, Dai Co Viet, Hai Ba Trung, Hanoi, Viet Nam
Received: May 26, 2017; Accepted: November 03, 2017

Abstract

We investigate the problem of extracting a subset of maximal cliques inside an undirected graph. This problem is considered as a NP-complete problem. In this work, we propose a heuristic algorithm that treats the above problem. In our algorithm, undirected graph is represented using the adjacency-list. Next, this representation is transformed into a new form so-called transaction database that is very familiar in data mining domain. Based on the new representation, we are able to count the frequency of subset of vertices inside undirected graph which is used to extract a set of maximal candidates that become possibly maximal cliques. Our experimental results show that this algorithm maintains a reasonable threshold in order to control its complexity.

Keywords: Maximal cliques, Graph mining, Heuristic algorithms, Apriori Algorithm

1. Introduction

In recent years, applications of graph mining get more and more interested by the datamining community. There are several reasons to explain these rising interests. Firstly, the graph seems a flexible way to represent visually many real-world problems. Secondly, graphs contain NP-problems that made more challenges to scientists. Among the graph NP-problems [1], we decide to choose the problem of extracting maximal cliques inside an undirected graph. The problem was found its many applications in various filed such as social network analysis [2], bioinformatics [3-4], information retrieval [5], computer vision [6], probabilistic graphical models [7-9]. All recent algorithms like [10-12] to solve the problem of listing all maximal cliques based on the recursive algorithm [13] reveal some inconvinences. These algorithms cannot run parallelly due to the dependence between recursive calls. In common cases, a deep recursion causes stack overflow that occurs while we used [12] to extract maximal cliques in quite dense graphs. We hope our proposed approaches will reduce theses inconvinences.

2. Preliminaries

We could present a social network by a simple undirected graph $G=(V,E)$ with $|V|=n$ vertices and $|E|=m$ edges. Without loss of generality, we suppose each vertex of G be labeled by an identical integer number. Each vertex corresponds to a node in the network and each edge represents a connection between two nodes. A maximal clique is well defined in graph theory (see some examples in Figure 1). Finding the complete list of all maximal cliques of a graph requires the algorithms with exponential complexity [10-14]. However, we need sometimes answer a question, given a certain graph, if we conduct a limited subset of listing maximal cliques inside the graph then a proposal algorithm will reduce possibly its time complexity requirement.

3. Reversible transformation

In order to choose an appropriate data structure to represent our graphs, we noticed the comment of [11] in which real-world graphs are usually sparse, i.e. $|E| = O(|V|)$ therefore the adjacency-list in [15] will replace the conventional adjacency-matrix in representing our undirected graph $G=(V,E)$. Recall that the adjacency-list representation of G requires only $O(|V|+2|E|)$ memory space, whereas the adjacency-matrix representation need much $O(|V|^2)$ memory space.

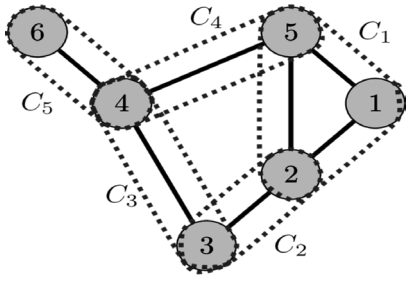
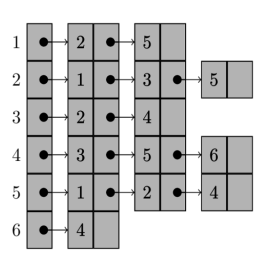


Fig. 1 A simple undirected graph G with $|V|=6$ and $|E|=7$ consists of five maximal cliques $C_1=\{1, 2, 5\}$, $C_2=\{2, 3\}$, $C_3=\{3, 4\}$, $C_4=\{4, 5\}$ and $C_5=\{4, 6\}$. Each maximal clique is closely contoured by a dotted cycle.

*Corresponding author: Tel.: (+84) 942.227.941
Email: phucta@soict.hust.edu.vn

Our idea is to make a reversible transformation between this adjacency-list representation and the transaction database in [16] then we can analyse our undirected graph based on counting co-occurrence frequencies of vertices through their adjacency list in this transaction database. In giving a new aspect for simple undirected graph representation, we hope to explorer its properties by using the Apriori algorithm of [17].



TID	Items
1	1, 2, 5
2	1, 2, 3, 5
3	2, 3, 4
4	3, 4, 5, 6
5	1, 2, 4, 5
6	4, 6

Fig. 2 Adjacency-list representation of G in Figure 1 is transformed into a transaction database on the right of this figure.

The table on the right in Figure 2 represents a transaction database containing a set of items $I=\{1,2,...,n\}$ and a set of transactions $T=\{t_1,t_2,...,t_n\}$ where t_i is subset of I with $i=1,2,...,n$. Each transaction t_i is associated with an identifier, called TID. Let R be a set of items. A transaction t_i is said to contain R if and only if R is a subset of t_i. An itemset that contains k items, is k-itemset. It is easy to prove that this transaction database requires $O(|V| + 2|E|)$ memory space the same as the adjacency-list representation.

We give several connections between a simple undirected graph $G=(V,E)$ and its transformed transaction database:

- An item in this transaction database corresponds to a vertex of G;
- An itemset in this transaction database corresponds to a set of vertices of G;
- Number of transactions in this transaction database is equal to the number of vertices $|V|$;
- Size of any transaction is at least 1, i.e. corresponding to an isolated vertex;
- Size of <*list of items*> corresponding to a vertex v is equal to $degree(v) + 1$.

4. Heuristic algorithm

Apriori algorithm [17] is used to mining frequent itemset, i.e. a set of items, in the transaction database DB. Each itemset R has its own occurrence frequency, called a *support* that is the number of transactions in DB containing this itemset. Given a transaction database DB and a minimum support threshold, *minsup*, the mining problem is to find the complete list of itemsets such that their supports are at least *minsup*.

Input : A simple undirected graph $G = (V, E)$ and a threshold *minsup*
Output: A set R of maximal cliques extracted from the undirected graph $G = (V, E)$

```
< a transformation of G = (V, E) into its transformed transaction database DB >;
C_1 ← init-pass(DB); F_1 ← {f | f ∈ C_1, support(f) ≥ minsup};
for k ← 2; F_{k-1} ≠ ∅; k++ do
    C_k ← candidate-gen(F_{k-1});
    foreach t_i ∈ T do
        foreach c ∈ C_k do
            if t_i contains c then
                increase support(c) by 1
            end
            if TID of t_i is an item of c then
                increase trans_counter(c) by 1
            end
        end
    end
    F_k ← {c | c ∈ C_k, support(c) ≥ minsup}
    R_k ← {c ∈ F_k | k = support(c) ∧ k = trans_counter(c)}
    R ← ∪_k R_k
end
return R
```

Fig. 3 Extract-Sub-Graph algorithm

Certainly, a maximal clique forms an itemset in this transaction database. If we can determine a suitable higher threshold *minsup*, extracting a set of maximal cliques is possible in the way that only some maximal cliques having frequency be superior to *minsup* are remained in the mining list (see Figure 3).

Firstly, unlike the recursive algorithms [10-13], the Apriori algorithm is a level-wise procedure, at k-level, it generates a set of candidates that are k-itemsets having probably frequency at least *minsup*. The algorithm gives us an opportunity to control the size of itemsets be extracted. For many applications, this key property of the algorithm makes an important advantage. Secondly, we need a subroutine to determine whether an itemset is formed by a maximal clique. Two following propositions will help us to make this subroutine

Proposition 4.1 *If C is a maximal clique of $G = (V, E)$ then its support number in the transformed transaction database is equal to its number of vertices.*

Proof:

Suppose the maximal clique $C=\{v_1,v_2,...,v_p\}$, where p is the number of vertices in C. Since every two vertices in C are adjacent, the adjacency-list of a vertex v_i must contain all other vertices v_j ($j\neq i$, $j=1,2,...,p$). The TID transaction corresponding to v_i then contains all vertices of the clique C. It means that C appears exactly once in each of p TID transactions corresponding to its vertices.

We'll prove by contradiction that *C* does not appear in any other TID transaction. For the sake of contradiction suppose *C* appear in a TID transaction corresponding to a vertex $u \notin C$. Then u is connected to all the vertices of C, and $C \bigcup u$ will form a clique. But this contradicts the fact that C is a maximal clique.

Eventually, we can conclude that *C* only appears exactly *p* times in the transformed transaction database that is equal to the number of its vertices.

End of proof

This proposition is a *necessary* condition to identify a maximal clique in the transformed transaction database, in opposite side, an itemset has its support number which is equal to its number of items is not surely formed by a maximal clique. For example, see the graph in the Figure 1, an itemset *{2,4}* has also its *support{2,4}=2* however two vertices don't form a maximal clique. We will call every itemset that satisfies the proposition 4.1's condition, be *maximal candidates*.

Proposition 4.2 *A maximal candidate M corresponds to a maximal clique if all its items are always found in each TID transaction corresponding to an item of M.*

Proof:

Since all items of *M* are always found in each TID transaction corresponding to an item of *M*, then every two vertices in the subgraph *C* associated with *M* are adjacent. It means *C* is a clique.

Suppose *C* is not a maximal clique, then there exists a maximal clique *U* containing *C*, i.e. $C \subset U$. Thus, the support number of *C* in the transformed transaction database should be at least equal to the number vertices of *U*, and then greater than the number of vertices of *C*. This contradicts the fact that *C* is a maximal candidate. In other words, *C* must be a maximal clique.

End of proof

This proposition is a *sufficient* condition to make sure an itemset be a maximal clique. For example, see again the transformed transaction database in the Figure 2, a maximal candidate *{4,5}* is scanned exactly once each of the TID=4 and TID=5 transactions then this maximal candidate is formed by a maximal clique C_4*={4,5}*.

After having all necessary theoretical propositions, we propose a heuristic algorithm, called Extract-Sub-Graph, to extract a set of maximal cliques from network in Figure 3.

Extract-Sub-Graph algorithm makes two principal changes, in line 1 and in line 16, in comparison to the Apriori algorithm [17]. In the line 1, there is a transformation of graph *G=(V,E)* into its transaction database. In the lines between 16 and 18, a subroutine is to check a k-itemset *c*, whether it satisfies both the propositions' 4.1 and 4.2 conditions, to become a maximal clique. Because the k-itemset *c* is extracted from F_k therefore *support(c)* is always superior to *minsup*. The most time complexity requirement command locating in line 4 of the algorithm [19] is to generate a set of candidates, like the Apriori algorithm, this command successively joins two previous k-1 itemsets into a new k-itemset for next iteration. The cardinality of a set of generated candidates becomes rapidly large, if the threshold *minsup* is assigned by a lower value. Consequently, it costly scans the whole database to check each candidate *c* is contained in each transaction t_i in *T*. Fortunately, if the threshold *minsup* gets a suitable higher value; the cardinality of this set is significantly reduced. This algorithm has same complexity as the Apriori algorithm [17] because we implemented two counters for each itemset *c*. A counter is to determine *support(c)* and the other for counting scanned times passed TID transaction in DB, it's called *trans_counter(c).* Based on the propositions 4.1 and 4.2, an itemset *c* is a maximal clique while its two counters are both equal to the size k of this itemset.

5. Implementation and expriments

We implemented the Extract-Sub-Graph algorithm in the C programming language, and ran experiments on a laptop with Ubuntu Client 64-bit, Intel Dual Core 1.6 Ghz processor and 4 GB of memory.

Table 1 A brief description of Mark NewMan's networks

network	|*V*|	|*E*|
karate	34	78
dolphins	62	159
lesmis	77	254
political books	105	441
adjnoun	112	425
football	115	612
celegensneural	297	1248
political blogs	1490	16715
netscience	1589	2742
astro-ph	16706	121251

Our experimental networks are curated by [18] (see Table 1) which consists of social networks such as relationships between friends in a karate club, savage dolphins in ocean, network of co-purchasing of political books by the Amazon's same buyers,

network of political blogs during the time of the 2004 presidential election, network of American football games, network of co-appearances of characters in Victor Hugo's novel "Les Misérables"etc

In all of these network data sets, especially when they have high number of vertices, the time requirement of running the Extract-Sub-Graph became high. In this case, we varied 5 different high values to assign the threshold *minsup* for each network data in order to decrease its numbers of generated candidates that causes long running time of the Extract-Sub-Graph algorithm. These generated candidates must be superior to *minsup* to become frequent itemsets (see Table 2).

Table 2. Elapsted times of expriments with 5 different high values of *minsup* using the Extract-Sub-Graph alg.

network	Elapsted times of expriments in miliseconds (ms)					
karate	minsup	5	4	3	2	1
	miliseconds	4493	7537	13451	16420	15278
dolphins	minsup	5	4	3	2	1
	miliseconds	24287	29391	53383	54010	69196
lesmis	minsup	10	9	8	7	6
	miliseconds	63840	70818	89488	108663	109116
political books	minsup	6	5	4	3	2
	miliseconds	180996	285100	358223	391730	402373
adjnouns	minsup	5	4	3	2	1
	miliseconds	154893	231953	334248	453702	452050
football	minsup	9	8	7	6	5
	miliseconds	397782	445517	479825	514039	554088
celegens-neural	minsup	8	7	6	5	4
	miliseconds	304638	307102	323506	314375	374021
political blogs	minsup	40	39	38	37	36
	miliseconds	490178	505132	508015	560902	593992
netscience	minsup	18	17	16	15	14
	miliseconds	154143	238825	264857	213925	223858
astro-ph	minsup	94	93	92	91	90
	miliseconds	309103	320117	483718	475804	500068

4. Conclusion

We have introduced the Extract-Sub-Graph algorithm for extracting a set of maximal full-connected subgraphs. An interesting point of this algorithm is that we not need a high-performance computer to realize a complex task of solving one of the NP-complete problems [1]. By determining a suitable threshold value, *minsup*, we chopped it up to a smaller problem that could be more easily solved. This idea gets a very practical use in a limited computer resource situation. Inside of this algorithm, we gave the undirected graph *G=(V,E)* representation under a transaction database form that help us to explorer graphical properties base on co-occurrence frequency among its vertices.

Acknowledgments

This research is funded by the Hanoi University of Science and Technology (HUST) under project number T2016-PC-041.

References

[1] Richard M. Karp, Reducibility Among Combinatorial Problems. In R. E. Miller and J. W. Thatcher (editors). Complexity of Computer Computations. New York: Plenum. (1971) 85–103.

[2] Harary, F., Ross, I.C. A procedure for clique detection using group matrix. Sociometry 20(3), (1957) 205-215.

[3] Grindley, H.M., Artymiuk, P.J., Rice, D.W., Willett, P.: Identification of tertiary structure resemblance in proteins using a maximal common subgraph isomorphism algorithm. J. Mol Biol. 229(3), (1993) 707 – 721.

[4] Koch, I.: Enumerating all connected maximal common subgraphs in two graphs. Theory Computer Science 250(1-2), (2001) 1–30.

[5] Koch, I., Lengauer, T., Wanke, E.: An algorithm for finding maximal common subtopologies in a set of protein structures. Journal Computer Biological 3(2), (1996) 289–306.

[6] Augustson, J.G., Minker, J.: An analysis of some graph theoretical cluster techniques. Journal ACM 17(4), (1970) 571–588.

[7] Hammersley, J. M.; Clifford, P. Markov fields on finite graphs and lattices (1971).

[8] Michael I. Jordan, Learning in graphical models. In lecture Statistic. Depart. Berkeley University (1999).

[9] Koller D., Friedman N., Probablistic Graphical Models: Principles and Techniques. MIT press (2009).

[10] Eppstein D., Loffter M., Strash D., Listing all maximal cliques in sparse graphs in near-optimal time. Exact complexity of NP-hard problems (2010).

[11] Eppstein D., Strash D., Listing all maximal cliques in large sparse real-world graphs. In proceedings Exprimental Algorithms, (2011) 364-375.

[12] Tomita E., Sutani Y., Wakatsuki M., A simple and faster Branch-and-Bound Algorithm for finding a Maximum clique with Computational Expriments. In Journal IEICE transaction, (2013) 1286-1298.

[13] Bron C., Kerbosh J., Algorithm 457 : Finding all maximal cliques of undirected graph. In journal Comm. (1973) 575-577.

[14] Tomita E., Sutani Y., Wakatsuki M., The worst-case time complexity for generating all maximal cliques and computational expriments. In Journal Theoritical Computer Science, (2006) 28-42.

[15] Cormen T., et al Introduction to Algorithms (3rd edition). MIT Press (2009).

[16] Han J., Kamber M., Datamining : Concepts and Techniques. Morgan Kaufman (2006).

[17] Agrawal R., Srikant R., Fast Algorithm for mining association rules in large database. Proceedings in the 20th international Conference on Very Large Database, (1994) 487-499.

[18] Newman MEJ, http://www-personal.umich.edu /~mejn/netdata/

[19] Pang-Ning Tan, Michael Steinbach, Vipin Kumar, Chapter6, Introduction to Data Mining, 1st edition, Pearson (2013).