A Parallel Algorithm for Enumerating All Maximal Cliques in Complex Network

Nan Du, Bin Wu, Liutong Xu, Bai Wang, Xin Pei
Beijing Key Laboratory of Intelligent Telecommunications Software and Multimedia,
School of Computer Science and Technology, Beijing University of Posts and Telecommunications,
Beijing 100876, China
{dunan,wubin,xliutong,wangbai}@bupt.edu.cn

Abstract

Efficient enumeration of all maximal cliques in a given graph has many applications in Graph Theory, Data Mining and Bioinformatics. However, the exponentially increasing computation time of this problem confines the scale of the graph. Meanwhile, recent researches show that many networks in our world are complex networks involving massive data. To solve the maximal clique problem in the real-world scenarios, this paper presents a parallel algorithm Peamc *(Parallel Enumeration of All Maximal Cliques) which exploits several new and effective techniques to enumerate all maximal cliques in a complex network. Furthermore, we provide a performance study on a true-life call graph with up to 2,423,807 vertices and 5,317,183 edges. The experimental results show that* Peamc *can find all the maximal cliques in a complex network with high efficiency and scalability.*

1. Introduction

Graph-based link mining is a new research field of the data mining technology, which emphasizes the interrelationship among different entities. Lots of researches now focus on the sub-structures [1,2,14] of a single graph to find some common relation patterns, among which the maximal clique is a very important one. Many existing algorithms for enumerating all maximal cliques, including *Bron*'s BK algorithm[8], *Sukyuama*'s algorithm[9], and *Kazuhisa*'s algorithm[10] involve the depth-first traversal of a search tree with specific pruning policies, while *Kose*'s algorithm[11] is based on the Apriori-like concepts. It takes advantage of the fact that every clique of size k, where $k \geq 2$, is comprised of two cliques of size k-1 that share k-2 vertices. However, the exponentially increasing time for the computation confines the scale of the graph and most of the above algorithms are based on the random networks. Recent researches indicate many real-world networks are complex networks which are large sparse graphs with the properties of short average path-length, power-law degree distribution and high clustering coefficient.

The main contribution of this paper is to provide a parallel algorithm *Peamc* for enumerating all maximal cliques in the large complex networks. Given a graph G with n vertices and m edges, in the worst case, *Peamc* runs with $O((\Delta \times M_c \times Tri^2)/p)$ time delay and in $O(n+m)$ space using p processing elements, where Δ is the maximum degree of G, M_c represents the size of the maximum clique and *Tri* denotes the number of all triangles in G respectively. The rest of the paper is structured as follows. Section 2 presents and analyzes the algorithm *Peamc*. Section 3 shows a systematic performance study of the experiments with real datasets taken from a mobile telecom carrier. Section 4 introduces the real world applications of *Peamc* and finally some conclusions and future work are provided in section 5

2. Algorithm *Peamc*

For the graph G, $V(G)$ and $E(G)$ denote the set of vertices and edges of G. For a vertex v, set $\tau(v)$ represents the neighbors of v. Since v has $|\tau(v)|$ neighbors, v and these neighbors could constitute $\alpha=|\tau(v)|\times(|\tau(v)|-1)/2$ triangles at most. We assume the actual number of triangles is β, then β/α denoted by ε is called the *clustering coefficient* of v and we use $\sigma(v)$ to store all the vertices that could constitute triangles with v and its neighbors. A complete sub-graph of G is called a *clique*. If a *clique* is not contained in any other *cliques,* this *clique* is called a *maximal clique*.

This work is supported by the National Science Foundation of China under grant number 60402011and the co-sponsored project of Beijing Committee of Education SYS100130422

2.1 Basic Idea and Method

Since that the triangle structure or 3-clique is a basic sub-structure of any clique whose size is larger than 3. It has a close relationship with the clustering coefficient property of the complex network. We could take advantage of this fact to design our traversal policy for the vertices which are contained in triangles by depth-first order, so that the maximal cliques can be detected with high efficiency. Because each search tree rooted with every vertex in G will be traversed, all the candidate cliques will thus be identified. To determine whether the candidate clique is a maximal clique, we present the following theorem.

Theorem 1. Let clique $S \subseteq V$, S is a maximal clique of G, if and only if for any $v \in S$ and $u \in \tau(v)\text{-}\tau(v) \cap S$, we have $S \not\subset \tau(u)$. □

Proof: Assume S is a maximal clique. If for any $v \in S$ and $u \in \tau(v)\text{-}\tau(v) \cap S$ such that $S \subset \tau(u)$. Then S is a clique with ε=1. Therefore, set $\{u\} \cup S$ is also a clique. However, we had S a maximal clique earlier, which means there is no superset of S being a clique, a contradiction. Assume for any $v \in S$ and $u \in \tau(v)\text{-}\tau(v) \cap S$, such that $S \not\subset \tau(u)$, which means at least there exists one vertex $t \in S$ such that $(u,t) \notin E$. Thus, for any $u \in \tau(v)\text{-}\tau(v) \cap S$, $\varepsilon \neq 1$ holds and S is a maximal clique. □

2.2 Pruning by Prediction

Peamc employs a pruning policy explained as follows.

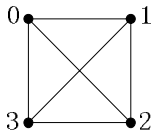

Figure 1. 4-clique

In Figure 1, we start from node 0 and choose node 1 from $\tau(0)$. After {0,1,2,3} is detected, we continue to choose node 2. However, we find {0,2,3} is not a maximal clique. This traversal contributes nothing. Consequently, we add a new set $\eta(v_0)$ that caches the maximal cliques obtained from the search tree rooted with v_0. After $\sigma(v_0)$ is calculated, we will first merge the current candidate clique with $\sigma(v_0)$ and check whether $\eta(v_0)$ contains the merged set. If so, the merged set will not be a maximal clique and the next traversal steps starting from the current candidate clique will be pruned. This pruning with prediction often improves the efficiency by a factor of 15% on average. This basic algorithm is summarized in Figure 2.

Input: a large sparse graph $G = (V, E)$;
Output: the complete set of all maximal cliques;
Method:
1: Read graph G
2: Generate set $\tau(v)$ for every vertex of G
3: `for` each vertex v of G
4: `call` *recursive_find_cliques*$(\{v\}, \tau(v))$
5: `end for`
Function: *recursive_find_cliques*(x, τ)
7: `for each` vertex $t \in \tau$ by the ascending order
8: calculate set σ
9: `if` $\sigma \neq \emptyset$
10: extend x with t
11: `call` *recursive_find_cliques*(x, σ)
12: `end if`
13: `else` // **Theorem 1**.
14: `end for`

Figure 2. Algorithm *Peamc*: the basic case

2.3 Parallel Model of *Peamc*

The enumeration of all maximal cliques is a NP problem and the basic algorithm requires each search tree rooted with every vertex of G to be traversed. Based on the fact that every search tree's traversal is independent with each other, to make the network that our algorithm can handle large enough in scale, we provide a parallel model of *Peamc*. Since that every vertex is accessed by the ascending order of its index, there will be no duplicate triangles and candidate cliques. Therefore, we have the following theorem.

Theorem 2. The enumeration of all maximal cliques in graph G with n vertices could be partitioned into n independent enumerations of all the maximal cliques from the search tree rooted with the correspondent vertices. □

Practically, the number of processing elements on the existing parallel platform P is often smaller than n. Thus a mapping scheme is required. We first sort the vertices of G by the descending order of their indices and define the partition ρ on a single processing element p as follows:

$$\rho = \left\{ v_i \middle| i = 0,1 \ldots \left\lfloor \frac{n}{P} \right\rfloor + 1 \right\} \quad v_i = u_j, u_j \in V, j = \begin{cases} i \times P + I_p, \frac{i}{2} = 0 \\ (i+1) \times P - p - 1, \frac{i}{2} \neq 0 \end{cases}$$

Let $\lambda = (\sum_0^{\left\lfloor \frac{n}{P} \right\rfloor + 1} |\tau(v_i)|) / (\lfloor n/p \rfloor + 1)$ .This mapping scheme gives most partitions a proper λ, which improves

the efficiency by a factor of 30% on average.

2.4 Analysis of *Peamc*

Based on the experimental results of our research, we find that the distribution of the maximal cliques whose size varies from 3 to M_c also holds a power-law feature, where M_c represents the size of the maximum clique. Let x denote the size of a maximal clique and y denote the number of the maximal cliques with size x. we have $y=kx^{-\alpha}$. For the traversal of the search tree rooted with vertex v_0, the calculation of σ from v_0 costs $\Delta\times[a_0\times(k\times3^{-\alpha})]^2$ time units $0\le a_0<1$, where Δ denotes the maximum degree of G. If $\sigma\neq\emptyset$, the calculation of σ' will cost $\Delta\times[a_1\times(k\times4^{-\alpha})]^2$ time units $0\le a_1<1$. If $\sigma'\neq\emptyset$, the calculation of σ'' will cost $\Delta\times[a_2\times(k\times5^{-\alpha})]^2$ time units $0\le a_2<1$. This process proceeds recursively until $\sigma=\emptyset$ Since M_c denotes the length of the deepest path from the root to the leaf, the time to enumerate all the maximal cliques by traversing a single search tree is represented as:

$$T_0=\Delta\times\left\{\left(a_0^{T_0}\times k\times3^{-\alpha}\right)^2+\left(a_1^{T_0}\times k\times4^{-a}\right)^2+\cdots+\left(a_{M_c-3}^{T_0}\times k\times M_c^{\ -\alpha}\right)^2\right\},\ a_i^{T_0}\in[0,1)$$

Because *Peamc* requires traversing each search tree rooted with every vertex of G, the total runtime T_S is defined as:

$$T_s=\sum_{i=0}^{n-1}T_i=\left\{\left(k\times3^{-\alpha}\right)^2\left[\left(a_0^{T_0}\right)^2+\cdots+\left(a_0^{T_{n-1}}\right)^2\right]+\cdots+\left(k\times M_c^{\ -\alpha}\right)^2\left[\left(a_{M_c-3}^{T_0}\right)^2+\cdots+\left(a_{M_c-3}^{T_{n-1}}\right)^2\right]\right\}\ \ a_i^{T_j}\in[0,1),\sum_{j=0}^{n-1}a_i^{T_j}=1$$

Since $\left[\left(a_0^{T_0}\right)^2+\cdots+\left(a_0^{T_{n-1}}\right)^2\right]<\left[\left(a_0^{T_0}\right)+\cdots+\left(a_0^{T_{n-1}}\right)\right]^2=1$,

then $T_s<\left\{\left(k\times3^{-\alpha}\right)^2+\cdots+\left(k\times M_c^{\ -\alpha}\right)^2\right\}<\Delta\times M_c\times\left(k\times3^{-\alpha}\right)^2$

Moreover, because all vertices of G are accessed by the ascending order of their index, which eliminates many duplicate results, and the pruning policy with prediction also reduces the search space greatly when the maximal cliques grow large, the actual runtime is much smaller than T_S. Thus, to enumerate all the maximal cliques in G, it will cost $O(\Delta\times M_c\times Tri^2)$, where $Tri=(k\times3^{-\alpha})$ The loading of G consumes $O(n+m)$ space. If we have n processing elements, according to theorem 2, it will cost $O((\Delta\times M_c\times Tri^2)/n)$ on average.

3. Experiment

To evaluate the performance of Peamc, we also implement *Bron*'s BK algorithm and *Kazuhisa*'s algorithm. Our experiments are done on a DAWN Cluster (84 3.2GHz processors with 2Gbytes of main memory on each node, Linux AS3). These algorithms have been first examined with random graphs. Given r and n, we build a graph with n vertices such that v_i and v_j are adjacent with probability 0.5 if $i+n-j$(mode n)$\le r$ or $i+n-i$(mode n)$\le r$. Then they are challenged with the complex networks and *Peamc* is also evaluated on large sparse call graphs built by real data taken from one telecom carrier for months in a city. The runtime in tables is expressed in seconds and we give the following notations for short: M.D. stands for Max Degree, M.C. for Maximal Cliques, Mc is the size of the maximum clique, Ka for Kazuhisa's Algorithm, BK for Improved BK, Pn for *Peamc* on n processing elements and s stands for speedup in the end.

Table 1. Comparison on random networks

G	r=10			R=30		
|V|	1000	2000	4000	1000	2000	4000
|E|	4534	8938	17987	14432	28709	58063
M.D.	16	16	24	64	82	92
M.C.	2239	4396	8894	19269	38060	78054
Mc	6	6	6	8	8	8
Ka	7	27	116	275	1006	12464
BK	0.5	4	29	0.7	4	31
P_1	0.2	0.4	0.9	29	56	123
P_{30}	0.01	0.02	0.04	1.69	3.27	6.49

Table 2. Comparison on complex networks

G	1	2	3	4	5
|V|	579	1161	2301	4557	8760
|E|	500	1000	2000	4000	8000
M.D.	43	44	45	57	81
M.C.	4	11	26	61	183
Mc	3	4	5	5	6
BK	0.112	0.740	6.032	42.22	313.5
Ka	0.027	0.052	0.105	0.208	0.403
P_1	0.0005	0.0014	0.0038	0.0083	0.0222

Peamc is also evaluated on large sparse call graphs built by real data from one telecom carrier for months in a city shown in Table 3.

Table 3. Results on 10 large sparse call graphs

G	1	2	3	4	5	6	7	8	9	10
|V|	512024	503275	540342	539299	543856	531444	529280	531861	562244	594186
|E|	1021861	900329	1030489	1014800	1034291	1020716	1012299	1007487	1060121	1152470
M.D.	673	731	980	2396	1355	913	1291	1134	1922	1310
M.C.	153362	118353	143259	139040	145569	148570	145809	144434	151039	168285
Mc	14	13	14	15	17	16	18	19	17	21
P_1	58	23	43	55	76	380	128	223	130	403
P_{30}	4	1	6	18	18	169	43	64	24	98

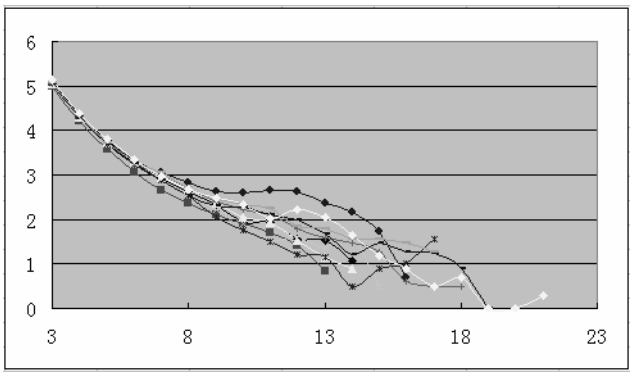

Figure 3 Distribution of the maximal cliques. The X axis represents the maximal clique size. The Y axis denotes the number of the maximal cliques in base-10 logarithm.

In Table 1, Improved BK's performance keeps stable regardless of $|E|$. While, in the case of r=30, as $|V|$ grows large, the clustering coefficient of every vertex is also increasing, thus *Peamc* outperforms Improved BK gradually. Our implementation of *Kazuhisa*'s algorithm is optimized for the spares graph according to his paper with $O(\Delta^4)$ time delay, *Kazuhisa*'s algorithm performs better in table 2 than in table 1. For the complex network, the distribution of the maximal cliques is confined by the power-law property shown in Figure 3. However the growth of the maximum degree of G does not have such limitation. Because most complex networks are large sparse graphs, our researches and experiments show that the growth of the number of the triangles is much slower than the growth of the maximum degree. Moreover, the size of the maximum clique is also much smaller than the maximum degree. Therefore, *Peamc* outperforms Kazuhisa's algorithm, which is shown in Table 2. In Table 3 we can find that the pruning policy with prediction becomes more efficient with groups having larger maximum cliques because the prediction becomes more accurate as the size of the maximal cliques grows large. The larger the maximum clique grows, the more efficient the pruning will become.

Furthermore, *Peamc* is challenged with another call graph of a month in a province with up to 2423807 vertices and 5317183 edges. We find 801381 maximal cliques using 70 processors in 26469 seconds and the size of the maximum clique is 26. Thus, we see that *Peamc* runs efficiently on the complex network.

4. Application

In our research, we use *Peamc* to enumerate all the maximal cliques in ten large sparse call graphs shown in Table 2 from graph 1 to 10. In Figure 4, we focus on the customer denoted by ▲. The gray filled circles represent the customers of the maximum 21-clique and those white blank circles denote their neighbors which are also involved in maximal cliques in the 10 months. We see that ▲ first appears in Jan and constitutes a triangle with other two persons in the quasi 19-clique (most pairs of its nodes are connected directly) which is part of the maximum 21-clique. ▲ has more connections in the following months indicating he or she gradually joins in the social circle of the 21 customers and becomes a new VIP customer to the telecom career. Figure 4 also shows that only 19 persons of the maximum 21-clique have appeared in Jan and only after May, another two persons denoted by the black filled circles join in the network. The degree of these two persons is increased sharply in Jun and keeps stable in the later months, which means these two persons are likely to know most of the 19 customers before and are recommended to use the same network with them, which is good to the telecom career's business. Moreover, we find that more and more other customers appearing as small white nodes inside the maximum 21-clique depend on this structure heavily and together they constitute a bigger quasi clique. On the contrary, Figure 5 gives the shrinking process of a maximal 9-clique from Mar to Jun. The shrinking of the maximal clique among the call graphs has a close relationship with the specific time and the composition of the structure. After one person in the 9-clique quitted in Apr, the whole structure shrank sharply in May. Thus, we can infer that this person holds an important position and has a heavy influence on the others

in the community of the 9 persons. If we can retain such kind of customer in advance, more lost profits will be prevented, which is another piece of good news to the telecom career.

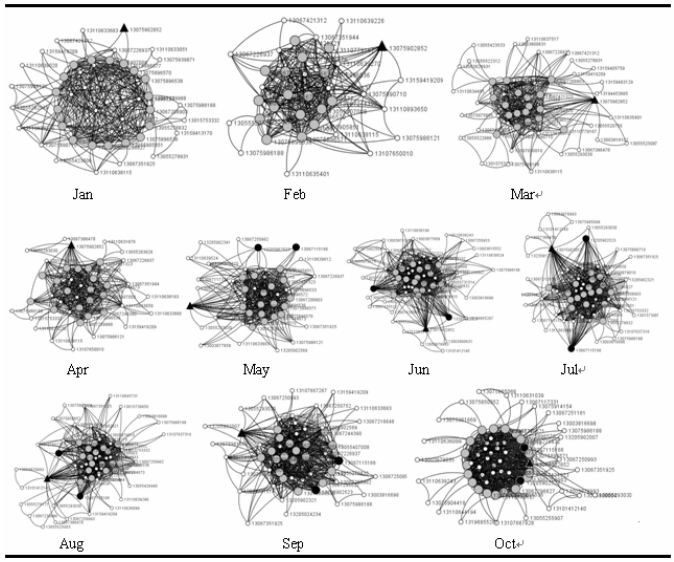

Figure 4. Maximum 21-clique evolution

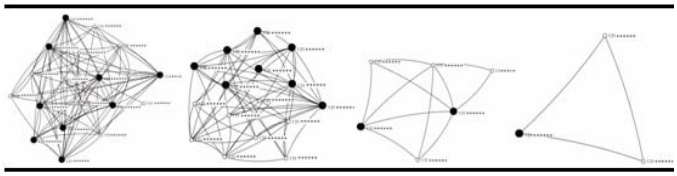

Figure 5. Maximal 9-clique shrinking

5. Conclusion and Future Work

In this paper, a novel parallel algorithm *Peamc* is provided, which exploits several effective techniques to efficiently enumerating all maximal cliques in complex network. Since most networks in our real world conform to the complex network model, our algorithm enjoys more attractive advantages in practice. A comprehensive performance study to compare *Peamc* with the existing algorithms on the real data sets has illustrated that *Peamc* is more efficient and scalable. Moreover, a promising application is shown to present and model the sub-structures' evolution and shrinking[12] among continuous sparse call graphs in the end.

To the future work, we will continue our research on interpreting the basic sub-structures of the complex network such as the quasi clique[14] and frequent sub-graphs, and focus on these structures' evolution by the time series analysis to have a better understanding of the network's dynamicity.

References

[1] R. Milo, Network Motifs: Simple Building Blocks of Complex Networks, Science (2002), 824

[2] M. E. J. Newman, The structure and function of complex networks, 2003 Society for Industrial and Applied Mathematics

[3] D.S. Johnson, On generating all maximal independent sets, Info. Proc. Lett., 27 (1988) 119 – 123

[4] T. Eiter and K. Makino, On computing all abductive explanations, Proc. AAAI '02, AAAI Press, pp.62 – 67, 2002

[5] R. Agrawal and R. Srikant, Fast algorithms for mining association rules in large databases, Proc. VLDB '94, 1994.

[6] Faisal N. Abu-Khzam, On the relative efficiency of maximal clique enumeration algorithms, with application to High-Throughput computational biology. Proceedings, International Conference on Research Trends in Science and Technology, Beirut, Lebanon, 2005

[7] Etsuji Tomita, The worst-case time complexity for generating all maximal cliques, COCOON 2004, LNCS 3106, 2004.

[8] C. Bron and J. Kerbosch, Algorithm 457: Finding all cliques of an undirected graph, Proceedings of the ACM, vol. 16, 1973

[9] S.Tsukiyama, A New Algorithm for Generating all the Maximal Independent sets, SIAM J. COMPUT. Vol. 6. 1977.

[10] Kazuhisa Makino, Takeaki Uno, New Algorithms for Enumerating All Maximal Cliques, SWAT 2004

[11] F. Kose, Visualizing plant metabolomic correlation networks using clique metabolite matrices, Bioinformatics, 2001

[12] Jure Leskovec, Jon Kleinberg, Christos Faloutsos, Graphs over Time: Densification Laws, Shrinking Diameters and Possible Explanations, KDD conference, 2005

[13] Kunal Punera, Joydeep Ghosh, CLUMP: A Scalable and Robust Framework for Structure Discovery, icdm conference 2005

[14] Jian Pei, Daxin Jiang, Aidong Zhang, On Mining CrossGraph QuasiCliques, KDD conference, 2005