Cliques are Too Strict for Representing Communities: Finding Large k-plexes in Real Networks

Alessio Conte[2], Donatella Firmani[1], Caterina Mordente[3], Maurizio Patrignani[1], and Riccardo Torlone[1]

[1] Roma Tre University, donatella.firmani@uniroma3.it,
{patrigna,torlone}@dia.uniroma3.it
[2] National Institute of Informatics, Japan, conte@nii.ac.jp
[3] Be Think Solve Execute, c.mordente@be-tse.it

Abstract. k-plexes are a formal yet flexible way of defining communities in networks. They generalize the notion of cliques and are more appropriate in most real cases: while a node of a clique C is connected to all other nodes of C, a node of a k-plex may miss up to k connections. Unfortunately, computing all maximal k-plexes is a gruesome task and state-of-the-art algorithms can only process small-size networks. In this paper we propose a new approach for enumerating large k-plexes in networks that speeds up the search by several orders of magnitude, leveraging on efficient techniques for the computation of maximal cliques.

1 Introduction

In the vast majority of networks representing real-world scenarios the distribution of edges is not uniform and it is often possible to clearly distinguish groups of nodes that are highly connected. The automatic detection of these groups, often called *communities*, helps to discover fundamental properties of large networks in a variety of different domains. For this reason this problem has been largely investigated [14]. A *clique* is a set of nodes in a network with all possible edges among them, and is a formal and strict way of defining a community. So strict, in fact, that cliques are generally thought to be too rigid to be used in practice [17]. A more appropriate notion in many practical cases is the *k-plex*: a set of nodes such that each of them has edges with all the others, with the possible exception of up to k missing neighbors (including itself). So, for example, for $k = 1$, k-plexes are cliques, for $k = 2$, each node may miss one edge, etc. Hence, k-plexes are a simple and intuitive generalization of cliques.

The problem of finding k-plexes arises in social network analysis [4], but it has wider applicability in several important areas employing graph-based data

SEBD 2018, June 24-27, 2018, Castellaneta Marina, Italy. Copyright held by the author(s).

mining [19]. Unfortunately, the detection of all maximal k-plexes is unpractical being hindered by two main problems: (i) maximal k-plexes are even more numerous than maximal cliques, even if most k-plexes are small and not significant; (ii) the most efficient algorithms in the literature, such as [6], can only be used on small-size graphs: our experiments shows that the largest networks we were able to analyze with such algorithms have a few hundred nodes. In this paper we propose a solution to the first issue that is also a solution to the second one. Namely, if we restrict the search to *large* k-plexes, which are the most meaningful in practice, we can devise efficient algorithms to detect them.

Indeed, computing all maximal k-plexes does not make sense when the purpose is that of detecting communities. In this respect, it is useful to focus on the relationship between s, the size of a k-plex, and k itself. Starting from $k = 1$, which corresponds to cliques, if we increase the value of k, we obtain progressively sparser communities that are clearly less interesting in practice. (For $s \leq k$, a k-plex can be composed of isolated nodes, and there exist many disconnected k-plexes for $s < 2k$.) In this framework, our strategy for finding large k-plexes relies on two main observations. First, the complexity of the problem can be reduced in the vast majority of cases on the basis of certain properties of large k-plexes that can be efficiently checked and that allows us to filter out a large portion of the network before starting their search. The second consideration is that, differently to what happens for k-plexes, the state-of-the-art techniques to compute all maximal cliques are able to scale up to millions of nodes by decomposing the network into small blocks [12]. Unfortunately, the decomposition approach cannot be easily adapted to the detection of k-plexes. However, we demonstrate that we can find **all k-plexes non-smaller than** m by looking in the neighborhood of cliques of a size that depends on k and m. The knowledge of cliques in a network provides a hint for finding all the significant k-plexes.

Structure of this paper. Section 2 contains an informal overview of our approach. Detailed description of our algorithms and their theoretical basis can be found in [10]. Main experimental results of [10] are reported in Section 3. Finally, Sections 4 and 5 contain related work and our concluding remarks.

2 Overview

Our approach is based on two main ideas: (i) before starting the search of k-plexes, we can filter out a relevant portion of the network in which necessary conditions for the presence of large k-plexes do not hold, and (ii) in large networks, cliques can drive the search of k-plexes. While the first point provides an effective way to simplify the problem at hand, the second can lead to an efficient strategy for finding k-plexes. Let us elaborate on these ideas starting with the problem of finding all k-plexes of *maximum* size. Assume that we have computed all the maximal cliques of a network and let ω be the size of the maximum clique. Then, a maximum k-plex has size at least ω, since cliques are also k-plexes. For

We recall that the problem of enumerating maximal k-plexes is NP-Hard.

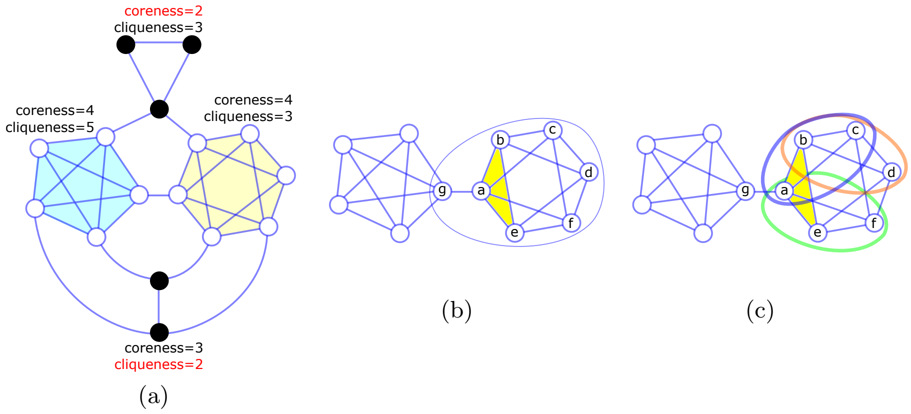


Fig. 1: An example network.

example, suppose we are searching for k-plexes in the network in Fig. 1a, which we will use as a running example in this section: we have that $\omega = 5$, since the maximum clique (the blue subgraph on the left hand side) involves five nodes.

Filtering. At this point, we observe that two filtering criteria can be applied.

1. CORENESS. Our first intuition follows from the very definition of k-plex: all the nodes of a k-plex of size m must have degree non-smaller than $m - k$. If we know that the size of a maximum k-plex is at least ω, this means that we can iteratively filter out any node that has degree lower than $\omega - k$. This corresponds to computing the *coreness* of all the nodes of the network (Lemma 2 of [10]), a process that can be executed in linear time [5]. For example, suppose we are searching for 2-plexes in the running example of Fig. 1a for which $\omega = 5$: we can filter out the three black nodes on the top of the picture since they have coreness 2, which is less than $\omega - k = 3$. In larger networks, this criterion allows us to cut up to 99% of the nodes.
2. CLIQUENESS. The second intuition is that any node of a k-plex of size m must be included in a clique of a size that depends on m. This is confirmed by Corollary 5.5 of [10] stating that any node of a k-plex larger or equal to m is included in a clique of size at least $\lceil m/k \rceil$. Then, if the size of the maximum k-plex is at least ω, we can cut out all nodes that do not belong to any clique of size at least $\lceil \omega/k \rceil$. For example, if we are searching for 2-plexes in the network depicted in Fig. 1a, we can filter out all nodes that do not belong to cliques of size at least $\lceil 5/2 \rceil = 3$, that is, the pair of black nodes in the bottom of the network. In large instances in our experiments this criterion, can be tested efficiently and is able to cut up to 98% of the nodes.

Even if some nodes can be filtered out both because their low cliqueness and low coreness, the network in Fig. 1a shows that the two filtering criteria are indeed independent. When both criteria are applied, the size of the network is reduced of magnitude and standard k-plexes algorithms may become feasible.

Once we have found the largest k-plexes, we may be interested into searching smaller ones. We have noted in the introduction that an exhaustive search does

Graph	n	density	ω	type
jazz	198	$1.41 \cdot 10^{-1}$	30	collaboration
grQc	5.241	$1.05 \cdot 10^{-3}$	44	collaboration
geom	6.158	$6.28 \cdot 10^{-4}$	22	collaboration
advogato	7.418	$1.75 \cdot 10^{-3}$	19	collaboration
hepPh	12.006	$1.64 \cdot 10^{-3}$	239	collaboration
astroPh	18.771	$1.12 \cdot 10^{-3}$	57	collaboration
newm	22.015	$2.42 \cdot 10^{-4}$	3	collaboration
mathSci	391.529	$1.14 \cdot 10^{-5}$	25	collaboration
dblp	511.163	$1.43 \cdot 10^{-5}$	115	collaboration
patents	3.8 M	$2.32 \cdot 10^{-6}$	11	citation

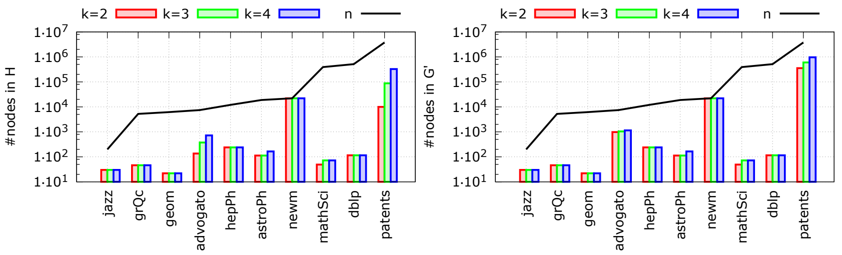
Table 1: Real-world networks in our experiments. Networks are sorted by size n.

not make much sense, since very small k-plexes are not significant, to the point that they may be even disconnected or composed by a set of isolated nodes. Hence, the second problem we tackle is to find all maximal k-plexes in the network of size bigger than a threshold m.

Local search. As mentioned above, our idea is to start from cliques, which are k-plexes but not necessarily maximal, and possibly enlarge them to find maximal k-plexes. Building on the cliqueness criterion, which ensures that each node of a k-plex C of size s is included into a clique of size at least $\lceil s/k \rceil$, we start from each of such cliques K. If we set $m \geq k^2$, we have that $|K| \geq \lceil s/k \rceil > k$, which implies that any other node of C must be adjacent to at least one node of K (in other words, K is a *dominating set* of C). Hence, we can search for C restricting to a block including K and all its adjacent nodes. For example, suppose you are searching for all maximal 2-plexes of size at least 5 in the network in Fig. 1a. Consider any clique of size at least $\lceil s/k \rceil = \lceil 5/2 \rceil = 3$, for example the clique $K = \{a, b, e\}$ (yellow triangle in Fig. 1b). The nodes of any k-plex of size at least 5 containing K are adjacent to K (surrounded nodes of Fig. 1b). We further reduce the size of the block by proving that C can be obtained by considering only nodes belonging to K and to other cliques of size at least $\lceil s/k \rceil$ intersecting with K (Lemma 5 of [10]). For example, the rightmost 2-plex of size 6 of Fig. 1a is all contained into the clique $\{a, b, e\}$ and three other cliques of size 3 intersecting with K (surrounded nodes of Fig. 1c). This gives rise to an efficient searching algorithm that decomposes the network into **blocks** each composed of one clique as the core, and all intersecting cliques as the boundary. Each block can be separately processed, possibly in a distributed environment.

3 Experiments

In this section, we compare our and previous algorithms over different real-world networks, and show the advantages and limitations of our approach. We use the algorithm in [12] for enumerating all the maximal cliques of the input graph G, and the algorithm in [6] for enumerating all the maximal k-plexes of the targeted graph **block**. Our code is publicly available [1]. We considered a mix of



(a) (b)

Fig. 2: (a) Number of nodes (log scale) in the sub-graph H, setting $m = \omega$. (b) Number of nodes (log scale) surviving the CORENESS criterion.

real-world networks with various sizes and characteristics. All our networks are publicly available on the LASAGNE meta-repository [2], and come from different human activities. The networks considered in our experiments, together with their number of nodes (n), density, maximum clique size (ω), and their type, are listed in Table 1 We compare the execution of our methods for enumerating *large* k-plexes, with the most recent method for enumerating *all* k-plexes of the input graph [6]. All our executions have a reasonable 6 hours timeout, after which they are interrupted.

Filtering. In Figure 2a, we show the number of nodes in the residual sub-graph H, resulting from the application of CORENESS and CLIQUENESS criteria. We use different values of k and $m = \omega$, i.e, the maximum clique size. As frame of comparison, we show the number of nodes in G (i.e., n). The figure shows that, except for `newm` where $\omega = 3$, the sub-graph H is order of magnitudes smaller than G. Notably, for different networks (`jazz`, `geom`, `hephPh`, `newm`, and `dblp`), H is left with only the nodes of the maximum k-plex. In such lucky cases we can even skip the execution of the enumeration step. Since the sub-graph produced by a given k is included in the sub-graph produces by $k + 1$, is not surprising that higher values of k yield more nodes in H. (Remember that for $k = n$ we have $G = H$.) However, for most instances, the sub-graph is small with respect to G (thus allowing for faster enumeration of k-plexes) for different values of k. Finally, our criteria have little impact on the `newm` network, because its maximum clique size is only 3. Figure 2b reports the number of nodes of G residual after the CORENESS criterion, applied separately. For the considered networks, most nodes are filtered out by CORENESS. Then, the structures that are too connected to be filtered by CORENESS but too small to play a role in the search for k-plexes non-smaller than ω, are filtered out by CLIQUENESS. Such an additional CLIQUENESS step has bigger impact in `advogato` and `patents`.

Maximum k-plexes. In Table 2, we show running times for different steps of our algorithm `max_plexes`(G, k) for finding the **maximum k-plex**, compared to the time required for enumerating all k-plexes (column "FULL ENUM"), over the same input graph. For this experiment, we set $k = 2$ and report in column

GRAPH	FULL	OUR APPROACH		
	ENUM	CORE	CLIQUE	ENUM
jazz	2 h	0,008 s	0,091 s	no need
grQc	> 6h	0,001 s	0,026 s	1,99 s
geom	> 6h	0,001 s	0,086 s	no need
advogato	> 6h	0,003 s	0,234 s	1 h 45 m
hepPh	> 6h	0,001 s	0,421 s	no need
astroPh	> 6h	0,046 s	0,892 s	2,92 s
newm	> 6h	0,22 s	0,167 s	> 6h
mathSci	> 6h	0,009 s	2,535 s	0,11 s
dblp	> 6h	0,005 s	4,336 s	no need
patents	> 6h	1,148 s	63,258 s	> 6h

Table 2: Running time for finding all the largest 2-plexes.

- CORE: running time of CORENESS criterion, and computing a sub-graph G';
- CLIQUE: running time of CLIQUENESS criterion over G', and computing H;
- ENUM: the execution time of the enumeration step over H.

The time for computing our criteria has little impact on the overall running time, which is dominated by the enumeration of 2-plexes of the residual sub-graph when necessary. As a consequence, in all the networks where H is left with only the nodes of the maximum clique, which is in turn also the maximum 2-plex, the computation of our algorithm ends successfully after fractions of seconds. For such networks, we write "no need" in the "ENUM" column. As frame of comparison, full enumeration of 2-plexes (i.e., as in [6]) requires hours even on our smallest network (jazz), and times out on the other networks. Whenever the enumeration step is necessary, its running time ranges from fractions of seconds (most networks) to 2 h, proportionally to the size of H (see Figure 2a for comparison). The only two instances that require more than 6 h are indeed nemw and patents, that correspond to the top two largest filtered sub-graphs. We observed that the results for higher values of k, namely $k = 3$ and $k = 4$, are similar. This is because the sub-graph H produced with higher valued of k contains only few more nodes than the sub-graph produced with $k = 2$.

Large k-plexes. In Table 3, we show the overall running time of our algorithm large_plexes(G, k, m) for finding **all k-plexes non-smaller than** m on different networks in our dataset. For this experiment, we use different values of k and set $m = 0.8\omega_k$, where ω_k is the maximum k-plex size, as computed by max_plexes(G, k). The time required for enumerating all k-plexes (column "FULL ENUM") of such networks is always larger than our timeout (6 hours). The table also show the number of k-plexes returned (column "FOUND"). All the networks considered contain less than a dozen k-plex non-smaller than $0.8\omega_k$, which are quickly found by our algorithm in most cases. Note that in this experiment CORENESS and CLIQUENESS are applied with $m = 0.8\omega_k$, which is possibly different from the threshold than in Figure 2a (i.e., when $0.8\omega_k \geq \omega$).

GRAPH	k	FULL ENUM	OUR APPROACH	
			TIME	#FOUND
grQc	2	> 6h	4,65 s	3
	4	> 6h	2,7 s	1
astroPh	2	> 6h	5 h 44 m	10
	4	> 6h	> 6h	-
mathSci	2	> 6h	2,75 s	7
	4	> 6h	> 6h	7

Table 3: Time for finding all k-plexes larger than 80% of the maximum clique.

4 Related Works

In the field of network analysis, dense substructures in graphs (aka dense subgraphs) are associated with communities, or more in general sets of closely related elements [14, 17]. The problem of finding these substructures has been extensively studied for decades, and continues to be the object of cutting edge research. The simplest and most rigorous definition of dense subgraph is the clique, i.e., a subgraph in which all nodes are pairwise connected. Many algorithms for finding all maximal cliques have been developed, most of them being inspired to the Bron-Kerbosh algorithm [7], such as [13, 18] or to the more recent paradigm of *reverse search* [3], such as [15, 9, 11]. McClosky [16] performs a thorough study to devise exact algorithms for finding the largest k-plex, and heuristics for finding lower upper bounds on its size, exploiting co-k-plexes (i.e., k-plexes on the complement graph) and graph coloring techniques. The usability of the algrorithms for finding the largest k-plex is however limited to small networks, as the running time exceeds the hour for graphs with hundreds of nodes. Cohen et al. [8] give a generic framework for enumerating all maximal subgraphs with respect to hereditary and connected hereditary graph properties, i.e., properties that are closed with respect to induced subgraphs and connected induced subgraphs, respectively. Berlowitz et al. [6] apply the framework in [8], together with insights on the k-plex problem, to produce efficient algorithms for the enumeration of maximal k-plexes and maximal connected k-plexes, which are respectively hereditary and connected hereditary. Other quasi clique models include the one defined by Zhai et al. [19], that is a k-plex with additional connectivity constraint, and more that can be found in this survey [17].

5 Conclusions

We have proposed a novel approach to the enumeration of large k-plexes, a formal and meaningful way to define interesting communities in real-world networks that generalizes the notion of clique. In the future, we intend to further extend the applicability of our approach and tackle the problem of computing large k-plexes on real world networks with millions of nodes. Our future work also includes experimenting with a variety of networks coming from different domains, such as (but not limited to) biological networks, web graphs, and product copurchasing networks.

Bibliography

[1] `http://patrignani.dia.uniroma3.it/large-k-plexes`. [Online; accessed Feb-2017].

[2] `http://lasagne-unifi.sourceforge.net`. [Online; accessed Feb-2017].

[3] D. Avis and K. Fukuda. Reverse search for enumeration. *Discrete Applied Mathematics*, 65(1-3):21 – 46, 1996.

[4] B. Balasundaram, S. Butenko, and I. V. Hicks. Clique relaxations in social network analysis: The maximum k-plex problem. *Oper. Res.*, 59(1):133–142, Jan. 2011.

[5] V. Batagelj and M. Zaversnik. An o(m) algorithm for cores decomposition of networks. *CoRR*, cs.DS/0310049, 2003.

[6] D. Berlowitz, S. Cohen, and B. Kimelfeld. Efficient enumeration of maximal k-plexes. In *SIGMOD*, SIGMOD '15, pages 431–444. ACM, 2015.

[7] C. Bron and J. Kerbosch. Finding all cliques of an undirected graph (algorithm 457). *Commun. ACM*, 16(9):575–576, 1973.

[8] S. Cohen, B. Kimelfeld, and Y. Sagiv. Generating all maximal induced subgraphs for hereditary and connected-hereditary graph properties. *Journal of Computer and System Sciences*, 74(7):1147 – 1159, 2008.

[9] C. Comin and R. Rizzi. An improved upper bound on maximal clique listing via rectangular fast matrix multiplication. *CoRR*, abs/1506.01082, 2015.

[10] A. Conte, D. Firmani, C. Mordente, M. Patrignani, and R. Torlone. Fast enumeration of large k-plexes. In *KDD*, pages 115–124. ACM, 2017.

[11] A. Conte, R. Grossi, A. Marino, and L. Versari. Sublinear-space bounded-delay enumeration for massive network analytics: Maximal cliques. In *ICALP*, pages 148:1–148:15, 2016.

[12] A. Conte, R. D. Virgilio, A. Maccioni, M. Patrignani, and R. Torlone. Finding all maximal cliques in very large social networks. In *EDBT*, pages 173–184, 2016.

[13] D. Eppstein and D. Strash. Listing all maximal cliques in large sparse real-world graphs. In *SEA*, pages 364–375, 2011.

[14] S. Fortunato. Community detection in graphs. *Physics reports*, 486(3):75–174, 2010.

[15] K. Makino and T. Uno. New algorithms for enumerating all maximal cliques. In *SWAT*, pages 260–272, 2004.

[16] B. McClosky and I. V. Hicks. Combinatorial algorithms for the maximum k-plex problem. *J. Comb. Optim.*, 23(1):29–49, 2012.

[17] J. Pattillo, N. Youssef, and S. Butenko. *Clique Relaxation Models in Social Network Analysis*, pages 143–162. Springer New York, New York, NY, 2012.

[18] E. Tomita, A. Tanaka, and H. Takahashi. The worst-case time complexity for generating all maximal cliques and computational experiments. *Theor. Comput. Sci.*, 363(1):28–42, 2006.

[19] H. Zhai, M. Haraguchi, Y. Okubo, and E. Tomita. A fast and complete algorithm for enumerating pseudo-cliques in large graphs. *International Journal of Data Science and Analytics*, 2(3-4):145–158, 2016.