2018

Enumerating Top-k Quasi-Cliques

Seyed-Vahid Sanei-Mehri
Iowa State University, vas@iastate.edu

Apurba Das
Iowa State University, adas@iastate.edu

Srikanta Tirthapura
Iowa State University, snt@iastate.edu

Enumerating Top-k Quasi-Cliques

Abstract
Quasi-cliques are dense incomplete subgraphs of a graph that generalize the notion of cliques. Enumerating quasi-cliques from a graph is a robust way to detect densely connected structures with applications to bio-informatics and social network analysis. However, enumerating quasi-cliques in a graph is a challenging problem, even harder than the problem of enumerating cliques. We consider the enumeration of top-k degree-based quasi-cliques, and make the following contributions: (1) We show that even the problem of detecting if a given quasi-clique is maximal (i.e. not contained within another quasi-clique) is NP-hard (2) We present a novel heuristic algorithm KernelQC to enumerate the k largest quasi-cliques in a graph. Our method is based on identifying kernels of extremely dense subgraphs within a graph, following by growing subgraphs around these kernels, to arrive at quasi-cliques with the required densities (3) Experimental results show that our algorithm accurately enumerates quasi-cliques from a graph, is much faster than current state-of-the-art methods for quasi-clique enumeration (often more than three orders of magnitude faster), and can scale to larger graphs than current methods.

Disciplines
Computer Sciences | Databases and Information Systems | Electrical and Computer Engineering | Systems and Communications

Comments
This is a pre-print of the article Sanei-Mehri, Seyed-Vahid, Apurba Das, and Srikanta Tirthapura. "Enumerating Top-k Quasi-Cliques." *arXiv preprint arXiv:1808.09531* (2018). Posted with permission.

Enumerating Top-k Quasi-Cliques

Seyed-Vahid Sanei-Mehri
Iowa State University
vas@iastate.edu

Apurba Das
Iowa State University
adas@iastate.edu

Srikanta Tirthapura
Iowa State University
snt@iastate.edu

***Abstract*—Quasi-cliques are dense incomplete subgraphs of a graph that generalize the notion of cliques. Enumerating quasi-cliques from a graph is a robust way to detect densely connected structures with applications to bio-informatics and social network analysis. However, enumerating quasi-cliques in a graph is a challenging problem, even harder than the problem of enumerating cliques. We consider the enumeration of top-k degree-based quasi-cliques, and make the following contributions: (1) We show that even the problem of detecting if a given quasi-clique is maximal (i.e. not contained within another quasi-clique) is NP-hard (2) We present a novel heuristic algorithm KERNELQC to enumerate the k largest quasi-cliques in a graph. Our method is based on identifying *kernels* of extremely dense subgraphs within a graph, following by growing subgraphs around these kernels, to arrive at quasi-cliques with the required densities (3) Experimental results show that our algorithm accurately enumerates quasi-cliques from a graph, is much faster than current state-of-the-art methods for quasi-clique enumeration (often more than three orders of magnitude faster), and can scale to larger graphs than current methods.**

I. INTRODUCTION

Finding dense subgraphs within a large graph is a foundational problem in graph mining, with wide applications in bioinformatics, social network mining, and security. Much attention has been paid to the problem of enumerating cliques, which are complete dense structures in a graph, e.g. [1], [2], [3], [4], [5], [6]. Usually, however, dense subgraphs are not cliques. The requirement of complete connectivity among vertices of the graph is often too strict, and there maybe edges missing among some pairs of vertices, or the existence of some edges may not be captured during observation. For example, cliques were found to be overly restrictive in identifying cohesive subgroups in social network analysis [7], [8], and instead, dense subgraph models were preferred that did not require complete connectivity. A similar need was found in the analysis of protein interaction networks [9]. This has led to the definition of "incomplete dense structures" or "clique relaxations" that are dense subgraphs where not every pair of vertices is connected. Such definitions can lead to more robust methods for identifying dense structures in a graph. In addition to being of great practical importance, the study of clique relaxations is of fundamental importance in graph analysis.

In this work, we consider a type of clique relaxation called a *degree-based* quasi-clique in a graph. For a parameter $0 < \gamma \leq 1$, a m-vertex subgraph H of a graph $G = (V, E)$ is said to be a degree-based γ-quasi-clique (henceforth called as "γ-quasi-clique) if the degree of each vertex in H is at least $\gamma \cdot (m-1)$. Note that if $\gamma = 1$, the definition required H to be a clique. By increasing γ, it is possible to make a stricter threshold for a subgraph to be admitted as a quasi-clique. If $\gamma < 1$, it is possible for the subgraph to be missing some edges among its vertices and still be admitted as a γ-quasi-clique. Quasi-clique mining has been applied in many areas such as biological, social, and telecommunication networks. Specific examples include: detecting co-functional protein modules from a protein interaction network [10], clustering in a multilayer network [11], [12], and exploring correlated patterns from an attributed graph [13]. A γ-quasi-clique is said to be *maximal* if it is not a proper subgraph of any other larger γ-quasi-clique. We consider enumerating maximal quasi-cliques. This formulation reduces redundancy in the output by ensuring that if a quasi-clique Q is output, then no other quasi-clique that is contained in Q is also output. Note that a maximal quasi-clique may not be the largest (maximum) quasi-clique in the graph.

We consider top-k maximal quasi-clique enumeration, where it is required to enumerate the k largest maximal quasi-cliques in the graph[1]. There are a few reasons why enumerating top-k maximal quasi-cliques is better than enumerating all maximal quasi-cliques. (1) if we focus on the top-k, then the output size is no more than k quasi-cliques. Compare this with enumerating all maximal quasi-cliques in a graph, whose output size can be exponential in the size of the input graph. For instance, it is known that there can be as much as $\Omega(3^{n/3})$ maximal cliques in a graph, and hence there can be at least as many maximal quasi-cliques, since each clique is a γ-quasi-clique with $\gamma = 1$. (2) the largest quasi-cliques in a graph are often the most interesting among all the quasi-cliques. (3) the time required for enumerating top-k can potentially be smaller than the time for enumerating all maximal quasi-cliques.

A straightforward approach to enumerate top-k maximal quasi-cliques is to first enumerate all maximal quasi-cliques in G using an existing algorithm for quasi-clique enumeration such as QUICK [14], followed by extracting the k largest among them. This approach has the problem of depending on an expensive enumeration of all maximal quasi-cliques. If the number of maximal quasi-cliques is much larger than k, then most of the enumerated quasi-cliques are discarded, and the resulting computation is wasteful. It is interesting to know if there is a more efficient way to enumerate the largest

[1]Our methods can also be adapted to enumerate only those quasi-cliques whose size is greater than a given threshold

maximal quasi-cliques in G. In this work, we present progress towards this goal. We make the following contributions:

NP-hardness of Maximality: First, we prove that even the problem of detecting whether a given quasi-clique in a graph is a maximal quasi-clique is an NP-hard problem. This is unlike the case of cliques – detecting whether a given clique is a maximal clique can be done in polynomial time, through simply checking if it is possible to add one more vertex to the clique. Note that our result is not about checking maximum sized quasi-cliques – it was already known [15] that finding the maximum sized γ-quasi-clique in a graph is NP-complete, for any value of γ. Instead, our result is about checking maximality of a quasi-clique.

Algorithm for Top-k γ-quasi-cliques: We present a novel heuristic algorithm KERNELQC for enumerating top-k maximal quasi-cliques without enumerating all maximal quasi-cliques in G. Our algorithm is based on the observation that a γ-quasi-clique typically contains a smaller but denser subgraph, a γ'-quasi-clique, for a value $\gamma' > \gamma$. KERNELQC exploits this fact by first detecting "kernels" of extremely dense subgraphs, followed by expanding these kernels into γ-quasi-cliques in a systematic manner. KERNELQC uses the observation that for $\gamma' > \gamma$, it is (typically) much faster to enumerate γ'-quasi-cliques than it is to enumerate γ-quasi-cliques. Further, the resulting set of γ'-quasi-cliques can be expanded into γ-quasi-cliques more easily than it is to construct the set of γ-quasi-cliques starting from scratch.

Experimental Evaluation: We empirically evaluate our algorithm on large real-world graphs and show that KERNELQC enumerates top-k maximal quasi-cliques with high accuracy, and is orders of magnitude faster than the baseline, which uses a state-of-the-art algorithm for quasi-clique enumeration. For instance, on the graph **`Advogato`**[2], KERNELQC yields a nearly 1000 fold speedup for enumerating the top-100 0.7-quasi-cliques, when compared with a baseline based on the QUICK algorithm [14].

While KERNELQC is not guaranteed to return exactly the set of top-k maximal quasi-cliques, it is very accurate in practice. Note that, given that the problem of even checking maximality of a quasi-clique is NP-hard, the cost of exact enumeration of maximal quasi-cliques is necessarily high. In many of the cases that we considered, the output of KERNELQC exactly matched the output of the exact algorithm that used exhaustive search. Usually, the error in the output, when compared with the output of the exact algorithm, was less than one tenth of one percent. See Section V for more details on the metrics used to measure the accuracy and performance of KERNELQC over the baseline algorithm. **Significantly, KERNELQC was able to scale to much larger graphs than current methods.**

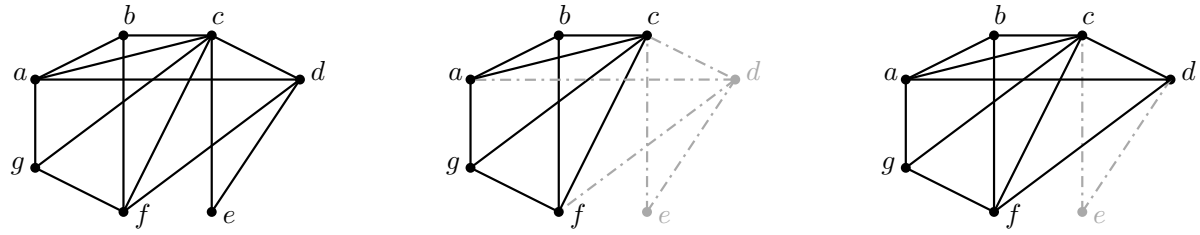
[2]details of the graphs used in the experiments are presented in Section V

A. Related Works

Degree-based Quasi-Clique: Motivated by a study on protein sequences, Matsuda et al. [16] first defined the degree-based γ-quasi-clique in the context of a protein sequence clustering problem. The degree based γ-quasi-clique has also been referred to as a γ-complete-graph in the literature [17]. Pei et al. [18] study the problem of enumerating those degree-based γ-quasi-cliques from a graph database that occur in the every graph of the graph database. Zeng et al. [19] studied the same problem as Pei et al. but generalize in a sense that their algorithm enumerates degree based γ-quasi-cliques that occur in at least a certain number of graphs in the database. Note that the algorithms discussed so far can also enumerate all maximal γ-quasi-cliques. Liu and Wong [14] propose the QUICK algorithm for enumerating all maximal γ-quasi-cliques from a simple undirected graph that uses a number of pruning techniques, some from prior works, and some newly developed. Lee and Lakshmanan [20] study the problem of finding a maximum γ-quasi-clique containing a given subset of vertices S of the original graph, and propose a heuristic algorithm. Recently, Pastukhov et al. [15] study the maximum degree-based γ-quasi-clique problem. First they prove that finding a maximum γ-quasi-clique is an NP-Hard problem, and present algorithms for a γ-quasi-clique of maximum cardinality. Note that while this work focuses on finding a single quasi-clique of the largest size, our goal is not just to find a single large quasi-clique, but to enumerate the k largest maximal quasi-cliques. Further, the NP-hardness result in [15] is for finding the maximum γ-quasi-clique, while our NP-hardness result is for finding if a quasi-clique is maximal.

Abello et al. [21] first study the problem of finding a density-based δ-quasi-clique, defined as a subgraph Q of the original graph with the ratio of the edges in Q to the total number of edges in a complete subgraph of size Q is at least δ. Note that a degree-based quasi-clique is also a density-based quasi-clique, but the converse is not true. They propose a heuristic algorithm for finding a large δ-quasi-clique. Uno [22] considered density-based quasi-cliques, and proposed an algorithm for enumerating all δ-quasi-cliques, with polynomial delay. In another study, Pattillo et al. [23] prove that deciding whether there exists a δ-quasi-clique of size at least θ is an NP-Complete problem. Brunato et al. [24] defines a (γ, δ)-quasi-clique combining the minimum degree requirement of degree based γ-quasi-clique and minimum edge requirement of density based δ-quasi-clique. They propose a heuristic algorithm for finding a maximum (γ, δ)-quasi-clique. Recently, Balister et al. [25] derive the concentration bound on the size of the density based maximum δ-quasi-clique following the work of Veremyev et al. [26].

Other Works on Dense Subgraphs: The study of dense subgraphs has attracted a wide spectrum of research for many decades. There have been many works on complete dense subgraphs such as maximal cliques [1], [27], [3], [2], [4], [28], [29], [5], maximal bicliques [30], [31], [32]. There are many different types of incomplete dense subgraphs other than quasi-



(a) Graph G (b) Quasi clique (c) Maximal Quasi-clique

Fig. 1: γ-quasi-clique with $\gamma = 0.6$ and $minsize$ = 5. (b) vertices $\{a, b, c, f, g\}$ form a γ-quasi-clique. (c) vertices $\{a, b, c, d, f, g\}$ form a maximal γ-quasi-clique

clique such as k-core [33], [34], [35], [36], k-truss [37] etc. A k-core is a maximal connected subgraph such that each vertex in that subgraph has degree at least k and this subgraph is quite different from quasi-clique in the sense that the degree threshold in the k-core is an absolute threshold whereas the threshold in the quasi-clique (either degree threshold or density threshold) are relative thresholds, equal to a certain factor (γ) times the size of the subgraph. In a k-truss subgraph, each edge is contained in at least $(k-2)$ triangles. k-truss is different from the quasi-clique because the threshold in the k-truss is an absolute threshold. Other works on dense subgraphs different from quasi-clique subgraph includes densest subgraph [38], [39], [40], [41], triangle densest subgraph [42], k-clique densest subgraph [43], [44] etc. Similar to k-core, these subgraphs are based on an absolute threshold for the degree, rather than a relative threshold. Prior work on top-k dense subgraph discovery includes the work of Zou et al. [45] on enumeration of top-k maximal cliques from an uncertain graph, defined as the set of cliques with the k largest clique probabilities and the works of Balalau et al. [46] and Galbrun et al. [47] on the enumeration of top-k densest subgraphs.

II. Preliminaries and Problem Definition

Let $G = (V, E)$ be a simple undirected graph. Let $V(G)$ denote the set of vertices and $E(G)$ denote the set of edges of G. Let $d^G(u)$ denote the degree of vertex u in G. When the context is clear, we use $d(u)$ to mean $d^G(u)$. We use the following definition of degree-based quasi-cliques.

Definition 1 (γ-quasi-clique): For parameter $0 < \gamma \leq 1$, a vertex-induced subgraph Q of G is called a γ-quasi-clique if Q is connected and, for every vertex $v \in V(Q)$, $d^Q(v) \geq \lceil \gamma(|Q| - 1) \rceil$.

Note that when $\gamma = 1$, the above definition reduces to a clique. For a γ-quasi-clique Q, by the phrase "size of Q" and notation $|Q|$, we mean the number of vertices in Q. A γ-quasi-clique Q is called *maximal* if there does not exist another γ-quasi clique Q' such that $V(Q) \subset V(Q')$ and $|Q| < |Q'|$. See Figure 1 for an example of the above definition.

Problem 1 (Top-k γ-QCE): Given integer $k > 0$, a parameter $0 < \gamma \leq 1$, a simple undirected graph $G = (V, E)$, enumerate k maximal γ-quasi cliques from G that have the largest sizes, among all maximal γ-quasi-cliques in G.

Given $0 < \gamma \leq 1$, a simple undirected graph $G = (V, E)$, the γ-quasi-clique enumeration (γ-QCE) problem asks to enumerate all maximal γ-quasi cliques from G. If the value of γ is clear from the context we sometimes use "QCE" to mean γ-QCE.

Quick algorithm for QCE: The current state-of-the-art algorithm for QCE is Quick [14], which takes as input a set of vertices X, degree threshold γ, and enumerates all maximal γ-quasi-cliques that contain X. By setting X to an empty set, one can enumerate all maximal γ-quasi-cliques of G. Note that Quick may also enumerate non-maximal quasi-cliques which need to be filtered out in a post-processing step. We modify Quick such that it omits the check for maximality in emitting quasi-cliques i.e. it enumerates all γ-quasi-cliques instead of only maximal ones – we call this version of the Quick algorithm as QuickM. Non-maximal quasi-cliques are filtered out at a later step, while enumerating top-k-quasi-cliques.

III. Hardness of Checking Maximality of a Quasi-Clique

It is easy to deduce that γ-QCE is an NP-hard problem, since the problem of enumerating maximal cliques is a special case when $\gamma = 1$. However, QCE presents an even more severe challenge. We now prove that even determining if a given quasi-clique is maximal is an NP-hard problem. This is very different from the case of maximal cliques – checking a given clique is maximal can be done in polynomial time, by simply checking if there exists a vertex outside the clique that is connected to all vertices within the clique. If there exists such a vertex, then the given clique is not maximal, otherwise it is maximal.

Problem 2 (Maximality of a Quasi-Clique): Given a graph $G = (V, E)$, a γ-quasi-clique $X \subseteq V$, determine whether or not X is a maximal quasi-clique in G.

Theorem 1: Maximality of a Quasi-clique is NP-hard.

Proof: We prove NP-hardness by reducing the r-clique problem, that asks whether a given graph $G' = (V', E')$ contains a clique of size r, to the problem of checking maximality of a quasi-clique. This r-clique problem is NP-complete [48]. Given graph G' on which we have to solve the r-clique problem, construct a graph $G = (V, E)$ as follows (See Figure 2). Let $V = V' \cup X$ where X is a set of $2r^2 + r$ additional vertices. X consists of three parts – two sets A_1 and A_2, each of size r^2, and B, of size r. We construct edges in G as follows:

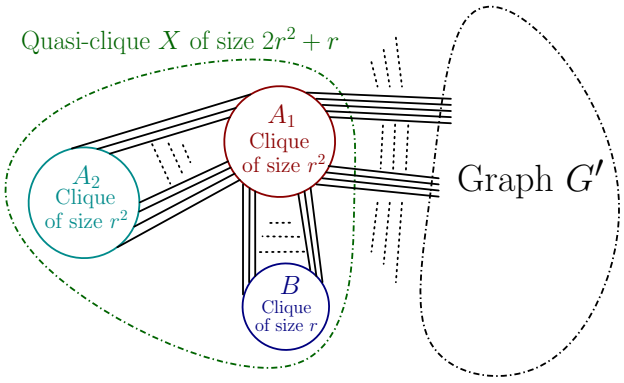


Fig. 2: Construction of graph G for Proof of Theorem 1.

- All edges E' in G' are retained in G
- Add edges within A_1, within A_2 and within B such that A_1 is a clique, A_2 is a clique, and B is a clique.
- Add edges connecting each vertex in A_1 with each vertex in A_2 and B. Thus, $A_1 \cup A_2$ is a clique and $A_1 \cup B$ is a clique, but $X = A_1 \cup A_2 \cup B$ is not a clique.
- Add edges connecting each vertex of A_1 to each vertex of V'.

Set $\gamma = \frac{r^2+r-1}{2r^2+2r-1}$. We first show that X is a γ-quasi-clique. To see this, consider that the total number of vertices in X is $(2r^2 + r)$. For X to be a γ-quasi-clique, each vertex should have a degree of at least $\lceil \gamma \cdot (2r^2 + r - 1) \rceil = \left\lceil \frac{(r^2+r-1)(2r^2+r-1)}{2r^2+2r-1} \right\rceil \leq (r^2 + r - 1)$. We can verify that every vertex in X has at least a degree of $(r^2 + r - 1)$.

We now claim that X is not a maximal γ-quasi-clique in G if and only if G' contains an r-clique.

(1) Suppose that G' contains an r-clique. There exists a set of vertices $L \subset V'$ such that L is a clique and $|L| = r$. Consider the set $Q = X \cup L$. We show that Q is a γ-quasi-clique. Since $X \cap L = \emptyset$, we have $|Q| = |X| + |L| = 2r^2 + 2r$. Therefore, $\lceil \gamma \cdot (|Q| - 1) \rceil = \left\lceil \frac{(r^2+r-1)(2r^2+2r-1)}{2r^2+2r-1} \right\rceil = r^2 + r - 1$. It can be verified that every vertex in Q has at least a degree of $r^2 + r - 1$. Thus, Q is a γ-quasi-clique.

(2) Suppose that X is not a maximal γ-quasi-clique in G. Then, there must be a non-empty set $M \subset V'$ such that $R = M \cup X$ is a γ-quasi-clique in G. We note that it is not possible that $|M| > r$. If this was the case, then the minimum degree threshold for a vertex in R is $\lceil \gamma \cdot (|R| - 1) \rceil = \left\lceil \frac{(r^2+r-1)(|R|-1)}{2r^2+2r-1} \right\rceil = \left\lceil \frac{(r^2+r-1)(2r^2+r+|M|-1)}{2r^2+2r-1} \right\rceil > r^2 + r - 1$, since $|M| > r$. However, the minimum degree of vertices in R is $r^2 + r - 1$ (consider a vertex from the set $B \subset R$).

Similarly, it is not possible that $|M| < r$. Let assume that was the case. Then, the minimum degree threshold for a vertex in R is $\lceil \gamma \cdot (|R| - 1) \rceil = \lceil \gamma \cdot (2r^2 + r + |M| - 1) \rceil$. However, the minimum degree of a vertex in R is $(r^2 + |M| - 1)$ (consider a vertex from M). It can be verified that since $|M| < r$, $\lceil \gamma \cdot (2r^2 + r + |M| - 1) \rceil > r^2 + |M| - 1$. Then, R cannot be a quasi-clique.

Therefore, it must be that $|M| = r$. In this case, M must be a clique of size r. In the case that M is not a clique, the minimum degree of a vertex in R is $r^2 + r - 2$ (consider a vertex from M). However, the minimum degree threshold for a vertex in R is $\lceil \gamma \cdot (|R| - 1) \rceil = \left\lceil \frac{(r^2+r-1)(2r^2+2r-1)}{2r^2+2r-1} \right\rceil = r^2 + r - 1 > r^2 + r - 2$. This completes the proof. ■

IV. Algorithm for Top-k QCE

In this section, we present algorithms for enumerating top-k γ-quasi-cliques given input graph G parameters k and γ.

A straightforward **baseline** algorithm for Top-k QCE is to enumerate all maximal quasi-cliques using the Quick algorithm [14], and then only output the k largest among them. We call this algorithm as Baseline.

KernelQC Algorithm: We next present our heuristic algorithm for Top-k QCE, KernelQC. The intuition is as follows. Our observations from experiments on a range of graphs showed that within a dense subgraph, a γ-quasi-clique for a given γ, there is usually a smaller, but denser subgraph, i.e. a γ'-quasi-clique with $\gamma' > \gamma$. We describe some results below.

We considered graphs **`Advogato`**, **`Route-views`**, and **`Bible`**[3], and $\gamma = 0.8$. We randomly sampled 1000 γ-quasi-cliques each from the graphs **`Advogato`**, **`Route-views`**, and **`Bible`**, of size at least 10 ($minsize = 10$)[4]. Interestingly, we found that **every sampled 0.8-quasi-clique from `Advogato`, `Route-views`, and `Bible` graphs had, as a subgraph, a γ'-quasi-clique of size at least 7, for different values of γ', ranging from 0.85 to 1.0.** Details are shown in the Fig. 3.

The above suggests that large γ-quasi-cliques (usually) contain γ'-quasi-cliques of substantial sizes as subgraphs for $\gamma' > \gamma$. Note that an adversary can form a γ-quasi-clique without any large γ'-quasi-clique contained within. However, our experiments show that this is not the case in real-world networks, and the size of γ'-quasi-cliques (or "kernels") in γ-quasi-cliques is relatively large (See Figure 3).

We further note that as γ increases, the complexity of finding γ-quasi-cliques decreases substantially. To see this, Figure 4 shows the computational cost of enumerating γ-quasi-cliques as γ increases. Note that the y-axis is in log-scale. The trend is that the cost decreases exponentially as γ increases.

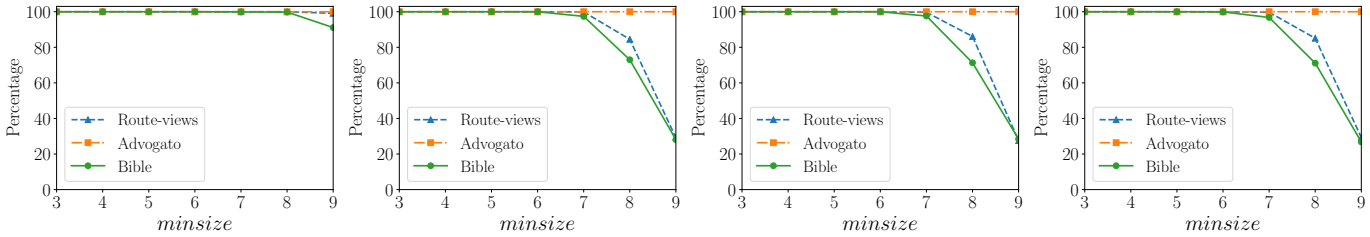
Based on the above observations, our algorithm idea is as follows. Given a threshold $0 < \gamma < 1$, we choose γ' such that $\gamma < \gamma' \leq 1$. We then enumerate the set Y consisting of the largest k' maximal γ'-quasi-cliques in the graph G. These dense subgraphs in Y are considered "kernels" that are then further expanded to recover k maximal γ-quasi-cliques in G. Thus, our algorithm has two parts:

(1) *Kernel Detection:* Find kernels in the graph, i.e. γ'-quasi-cliques for the chosen value of γ'. Then, among all kernels, largest k' maximal kernels are extracted.

(2) *Kernel Expansion:* Expand detected kernels into larger γ-quasi-cliques. This can be performed by iterating through the enumerated γ'-quasi-cliques and then using an existing algorithm for QCE, such as Quick [14] to enumerate

[3]These graphs are described in Table I

[4]We did not consider the graph **`Slash`** because the the size of largest γ-quasi-clique in this graph is less than 10.



(a) $\gamma' = 0.85$ (b) $\gamma' = 0.9$ (c) $\gamma' = 0.95$ (d) $\gamma' = 1.0$

Fig. 3: On the x-axis is the size of a γ'-quasi-clique. The y-axis shows what fraction of the sampled 0.8-quasi-cliques contained a γ'-quasi-clique of a given size. The results are shown for three different graphs, **`Advogato`**, **`Route-views`**, and **`Bible`**.

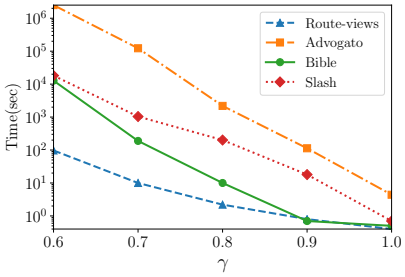


Fig. 4: Runtimes of γ-quasi-clique enumeration for different values of γ, for $minsize = 5$, and for different graphs.

all maximal quasi-cliques that contain each kernel. Next, among all extracted γ-quasi-cliques, largest-k maximal γ-quasi-cliques are enumerated.

The KERNELQC algorithm is described in Algorithm 1. In Line 2 of Algorithm 1, a modified version of QUICK, QUICKM, is used to extract all γ'-quasi-cliques. This version does not actually check if a quasi-clique is maximal, before outputting it. For KERNELQC, the quasi-cliques from the subroutine need not be maximal, since Algorithm 2 sorts quasi-cliques in an ascending order of their sizes and suppresses non-maximal quasi-cliques. By omitting a maximality check, QUICKM is more efficient than QUICK.

Lemma 1: Algorithm 2 returns at most k largest quasi-cliques that are maximal with respect to a set of quasi-cliques S (i.e. not contained within any other quasi-clique in S).

Proof: A quasi-clique q is added to the set Q in Line 5 of Algorithm 2 if the size of Q is less than k and q is not a subset of any other quasi-clique in Q (Using the **`if`** block in Line 4). Because of the latter condition, all quasi-cliques in Q are maximal with respect to the quasi-cliques of S. On the other hand, since all γ-quasi-cliques in S are sorted in an ascending order of their sizes (Line 1), Q maintains the largest maximal quasi-cliques from S, and the size of Q cannot be larger than k. ■

With Lemmas 1 and 2, we show that every γ-quasi-clique, returned by Algorithm 1, is maximal and has at least $minsize$ vertices.

Lemma 2: The set R in Algorithm 1 contains at most k γ-quasi-cliques, where each quasi-clique has at least $minsize$ vertices, and is a maximal quasi-clique in the graph.

Proof: In Line 2 of Algorithm 1, QUICKM extracts all γ'-quasi-cliques in the graph G. Then in Line 3 and by Lemma 1, X contains largest maximal γ'-quasi-cliques of G, where $|X| \leq k'$. In Lines 5 and 6, every γ-quasi-clique of the set Z contains at least a γ'-quasi-clique from the set Y. It is because we expand every γ'-quasi-clique in Y by QUICKM using the parameter γ. Therefore, any quasi-clique in the set Z has at least $minsize$ vertices. Since $R \subseteq Z$ (Line 7), any quasi-clique in R has at least $minsize$ vertices. In addition, by Lemma 1, we know that $|R| \leq k$.

In the rest, we show that quasi-cliques in R are maximal in G. By contradiction, assume that there is a γ-quasi-clique in R which is *not* maximal in G, *i.e.* suppose that there are two γ-quasi-cliques h and h' in G s.t. $h \in R$, $h \subset h'$, and $h' \notin R$. We know that h is discovered by the expansion of a γ'-quasi-clique q (Line 6). Hence, $q \subseteq h$ and $q \subset h'$. On the other hand, QUICKM ensures that all γ-quasi-cliques containing q are enumerated. The enumerated quasi-cliques are added to Z in Line 6. Therefore, $h' \in Z$ because h' is a γ-quasi-clique and contains q. In addition, since $h \subset h'$, Lemma 1 ensures that $h' \in R$ and $h \notin R$. This contradicts our assumption that $h \in R$ and $h' \notin R$. ■

In Line 1 of Algorithm 1, we need to choose two user-defined parameters, γ' and k', based on the given values of γ and k. Here, we discuss the influence of these parameters on the accuracy and runtime of KERNELQC.

Dependence on γ': For a given γ, varying the value

Algorithm 1: KERNELQC $(G, \gamma, minsize, k)$

Input: Graph $G = (V, E)$, parameter $0 < \gamma < 1$, size threshold $minsize$, and an integer k.
Output: k maximal γ-quasi-cliques in G with at least $minsize$ vertices in each.

1 Choose γ' such that $\gamma < \gamma' \leq 1$, and $k' \geq k$
2 $X \leftarrow$ QUICKM$(G, \phi, \gamma', minsize)$ ▷ *Kernel Detection – retrieve γ'-quasi-cliques from G.*
3 $Y \leftarrow$ TOPKMAXIMALQC(X, k') ▷ *Algorithm 2.*
4 $Z \leftarrow \emptyset$
5 **for** *a quasi-clique* $q \in Y$ **do**
6 $\quad Z \leftarrow Z \cup$ QUICKM$(G, q, \gamma, minsize)$ ▷ *Kernel Expansion – add γ quasi-cliques through expanding q.*
7 $R \leftarrow$ TOPKMAXIMALQC(Z, k) ▷ *Algorithm 2.*
8 **return** R

Algorithm 2: TOPKMAXIMALQC (S, k)

Input: Set of quasi-cliques S and an integer k.
Output: top (largest)-k maximal quasi-cliques from S.

```
Sort S in an ascending order of sizes of quasi-cliques
Q ← ∅
for a quasi-clique q ∈ S do
    if (|Q| < k) ∧ (∀q' ∈ Q, q ⊈ q') then
        Q ← Q ∪ q
return Q
```

$\gamma' \in (\gamma, 1]$ has effect on the runtime of KERNELQC. Based on our observation in Figure 4, for a high value of γ', the kernel detection phase of KERNELQC can extract kernels faster. However, these kernels have relatively smaller sizes, and the expansion phase will take a longer time. Conversely, if γ' is small (close to γ), kernel detection takes more time to extract γ'-quasi-cliques while the kernel expansion phase requires less time as kernels have relatively larger sizes and less chance to be expanded.

Dependence on k': In Algorithm 1, k' decides the number of kernels, which need to be extracted in the kernel detection and then expanded to mine γ-quasi-cliques. The higher the value of k', the more the kernels are needed to be processed in KERNELQC. Then, the runtime of KERNELQC may increase with a higher value for k'. However, the chance to mine larger maximal γ-quasi-cliques also increases since more kernels need to be enlarged in the kernel expansion phase. Therefore, a higher value of k' can increase both the runtime and the accuracy of KERNELQC.

In Section V, we present an empirical sensitivity analysis of parameters $minsize$, γ, γ', k, and k' on the accuracy and runtime of the algorithm.

V. EXPERIMENTS

Networks and Experimental Setup. We used real-world networks from publicly available repository at KONECT.[5] The networks we used are summarized in Table I and are converted to simple graphs by removing self-loops and multiple edges. We implemented the BASELINE and KERNELQC algorithms in C++ and compiled with `g++` compiler with `-O3` as the optimization level. The experiments are conducted on a cluster of machines equipped with a 2.0 GHz 8-Core Intel E5 2650 and 64.0 GB memory.

Metrics. As discussed in Section IV, KERNELQC is a heuristic algorithm for extracting the top-k maximal γ-quasi-cliques. There is no guarantee that it will always be correct, i.e. it may not always enumerate the k largest maximal quasi-cliques. On the other hand, BASELINE mines the exact top-k maximal γ-quasi-cliques. We need a metric to measure the accuracy of KERNELQC compared to the BASELINE algorithm. For this purpose, we use Søergel similarity, which is as follows. Suppose that $H = \langle h_1, h_2, h_3, \cdots, h_k \rangle$ is an ascending ordered list, maintaining the sizes of k maximal

[5]http://konect.uni-koblenz.de/

Graph	#Vertices	#Edges	Maximum Degree	Average Degree
Advogato	5155	39,285	803	15.24
Route-views	6474	12,572	1458	3.88
Bible	1773	9131	364	10.30
Slash	51,083	116,573	2915	4.56
Live-mocha	104,103	2,193,083	2980	42.13
Youtube	1,134,890	2,987,624	28,754	5.27
Hyves	1,402,673	2,777,419	31,883	3.96

TABLE I: Summary of the input graphs.

γ-quasi-cliques returned by KERNELQC.[6] Similarly, suppose that $Z = \langle z_1, z_2, z_3, \cdots, z_k \rangle$ is a list in an ascending order, which contains the sizes of the top-k maximal γ-quasi-cliques, returned by BASELINE, the exact algorithm. The Søergel similarity between two lists H and Z is as follows:

$$\text{Søergel similarity (H, Z)} = \frac{\sum_{i=1}^{k} |h_i - z_i|}{\sum_{i=1}^{k} \max(h_i, z_i)} \times 100 \quad (1)$$

Using a similar method, we can compute the error percentage of a list H compared to a list Z. Here, we define how we measure the error percent of a list H from a list Z:

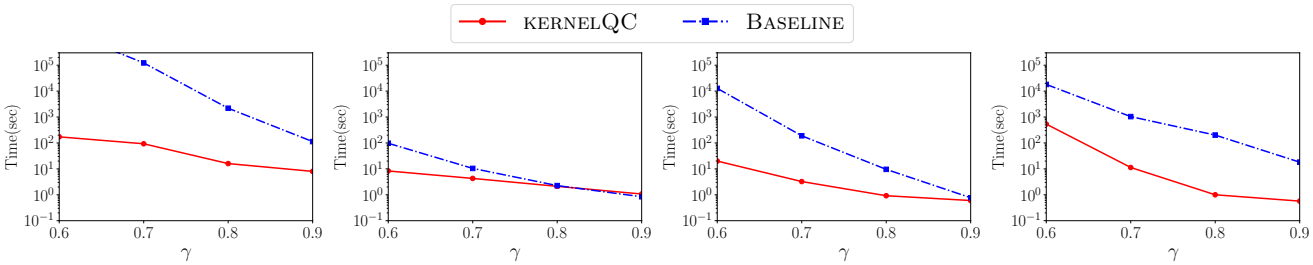
$$\text{Error percent (H, Z)} = \left(1 - \frac{\sum_{i=1}^{k} |h_i - z_i|}{\sum_{i=1}^{k} \max(h_i, z_i)}\right) \times 100 \quad (2)$$

For our purpose, the Søergel similarity is better suited than other metrics such as Jaccard similarity. In particular, if we used Jaccard similarity to measure the similarity between vertex sets of two quasi-cliques, this will fail to consider the sizes of the quasi-cliques. If two algorithms return sets of quasi-cliques that are of exactly the same sizes, but whose elements are different, then the Jaccard similarity will show a poor match, while the Søergel similarity will show a perfect match.

Note that Søergel similarity shows the similarity of two lists of numbers. Here, we consider the lists of *sizes* of quasi-cliques, obtained by KERNELQC and BASELINE algorithms. As mentioned above, the two lists with length k, where each list is sorted in an ascending order of sizes of quasi-cliques. Henceforth, when we refer to the error percent of KERNELQC, we compare the returned lists of KERNELQC and BASELINE using Equation (2).

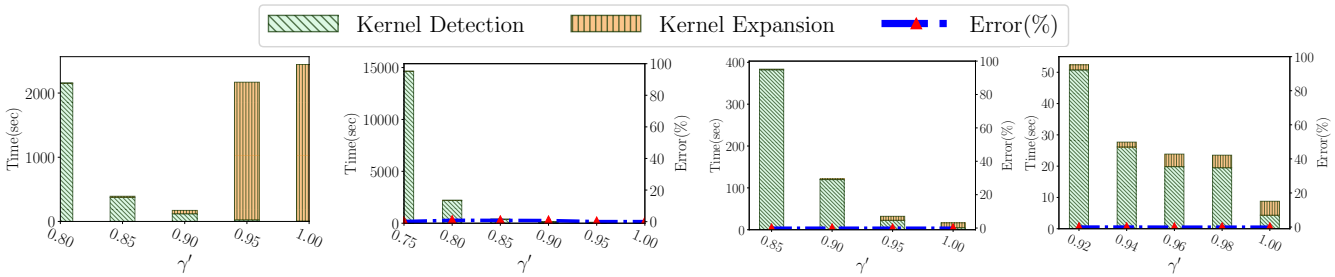
Runtime Compared to BASELINE: The experiments show that KERNELQC yields a significant speedup over the BASELINE, for enumerating Top-k-quasi-cliques. For example, in Figure 6b in graph **Advogato** with $\gamma = 0.7$, $k = 100$, and $k' = 300$, when we set $\gamma' = 0.9$, KERNELQC yields a speedup of 984x over BASELINE. For $\gamma = 0.6$, the speedup is even more since BASELINE did not finish after 259K secs while KERNELQC took only 172 secs.[7] For the graph **Bible**, with $\gamma = 0.6$ and $\gamma' = 0.8$, KERNELQC yields a 34x speedup over BASELINE (Figure 8a). In **Slash** and the same values for γ and γ', KERNELQC yields a 638x speedup over BASELINE (Figure 9a). On the **Route-views** graph, the speedup is not high, especially for large values of γ. The reason is that this graph is not very dense, and even the BASELINE had a small

[6]A size of a quasi-clique is the number of vertices in the quasi-clique.
[7]K stands for thousands



(a) **Advogato** (b) **Route-views** (c) **Bible** (d) **Slash**

Fig. 5: The runtimes of KERNELQC and BASELINE as a function of γ.



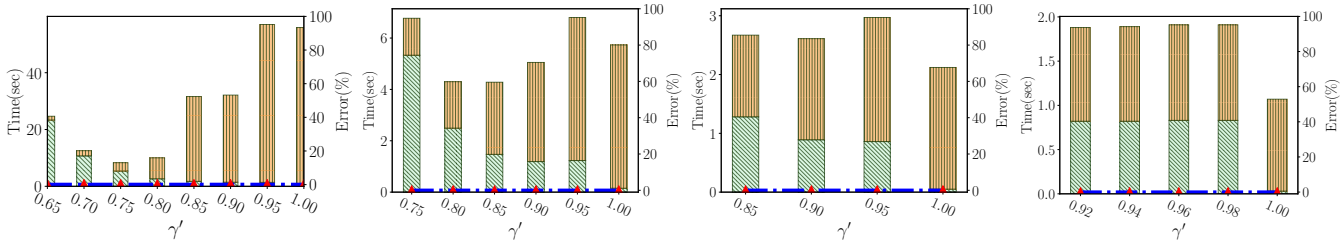
(a) $\gamma = 0.6$. BASELINE did not finish after 259,200 secs.

(b) $\gamma = 0.7$, error is 0%. BASELINE takes 122618 secs

(c) $\gamma = 0.8$, error is 0%. BASELINE takes 2185 secs

(d) $\gamma = 0.9$, error is 0%. BASELINE takes 113 secs

Fig. 6: **Advogato**, $k = 100$, $k' = 300$, $minsize = 5$.



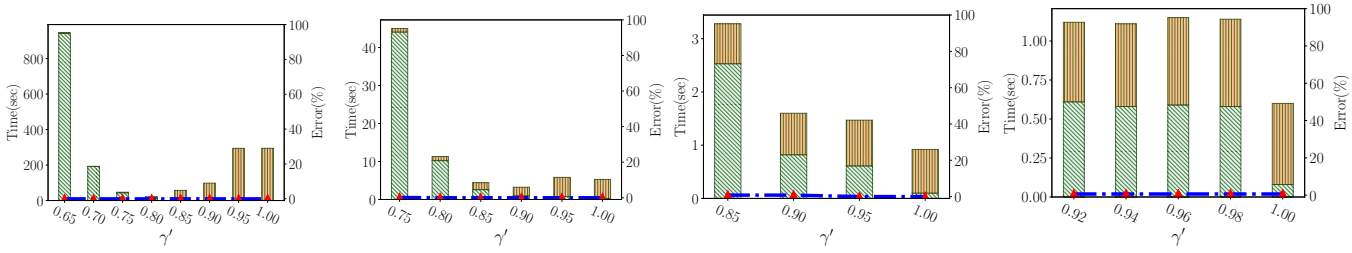
(a) $\gamma = 0.6$, error is 0%. BASELINE takes 97 secs

(b) $\gamma = 0.7$, error is 0%. BASELINE takes 10 secs

(c) $\gamma = 0.8$, error is 0%. BASELINE takes 2 secs

(d) $\gamma = 0.9$, error is 0%. BASELINE takes 0.9 secs

Fig. 7: **Route-views**, $k = 100$, $k' = 300$, $minsize = 5$.



(a) $\gamma = 0.6$, error is 0%. BASELINE takes 12760 secs

(b) $\gamma = 0.7$, error is at most 0.1%. BASELINE takes 189 secs

(c) $\gamma = 0.8$, error is at most 0.8%. BASELINE takes 10 secs

(d) $\gamma = 0.9$, error is at most 0.6%. BASELINE takes 0.8 secs

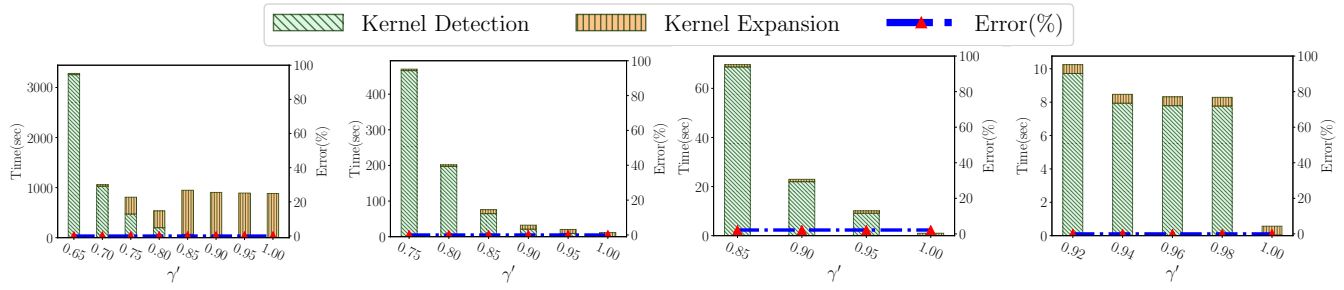
Fig. 8: **Bible**, $k = 100$, $k' = 300$, $minsize = 5$.

runtime (< 100 secs). For this graph, obtaining high speedups is not as important.

Error Rate: The results show that KERNELQC has a high accuracy for different graphs and various parameter settings while achieving a huge speedup over BASELINE. As shown in Figures 6–9, error percentage for most graphs and different values of γ and γ' is less than 0.9%, and is often zero (i.e. exactly matches with the output of BASELINE). The highest error among all experiments is 2.1% and belongs to the graph **Slash** where γ is 0.8 (Figure 9c). We did not report the error percent of Figure 6a because the BASELINE did not finish after 259K secs.

A. *Dependence on γ'*

In Figures 6–9, we ran KERNELQC for different values of γ' and γ on four graphs **Advogato**, **Route-views**, **Bible**, and **Slash**. The purpose of these experiments is to understand the effects of the user-defined parameters γ and γ' on KERNELQC in terms of accuracy and time-efficiency. These experiments are helpful to choose an optimum value for γ'



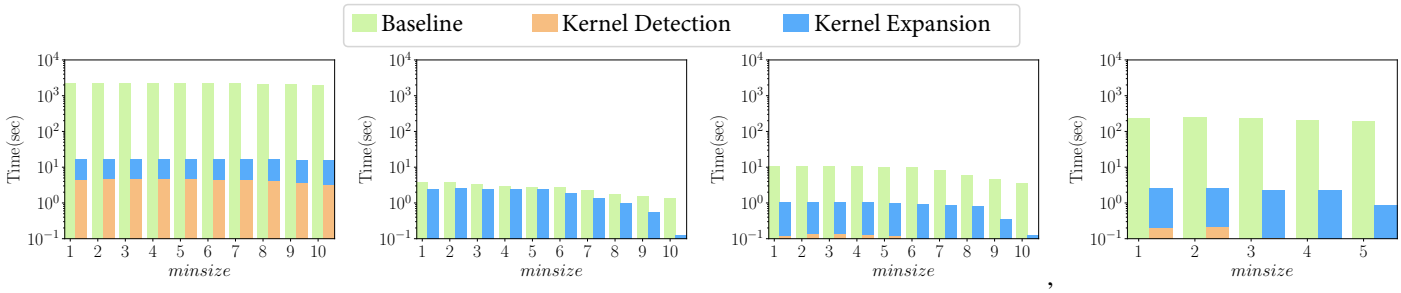
(a) $\gamma = 0.6$, error is 0%. BASELINE takes 18152 secs

(b) $\gamma = 0.7$, error is 0%. BASELINE takes 1039 secs

(c) $\gamma = 0.8$, error is at most 2.1%. BASELINE takes 201 secs

(d) $\gamma = 0.9$, error is 0%. BASELINE takes 18 secs

Fig. 9: **Slash**, $k = 100$, $k' = 300$, $minsize = 5$.



(a) **Advogato**

(b) **Route-views**

(c) **Bible**

(d) **Slash**

Fig. 10: The runtimes of KERNELQC and BASELINE as a function of $minsize$, for $\gamma = 0.8$, $\gamma' = 1.0$, $k = 100$, and $k' = 300$.

which can result in a good balance between accuracy and runtime. For all graphs, we set $k = 100$, $k' = 300$ to find Top-k quasi-cliques, and $minsize = 5$ which is the minimum size threshold of quasi-cliques.

Performance of kernel detection and expansion: Here, we show how different values of γ and γ' can have effect on two parts of KERNELQC. When γ' is close to γ kernel detection is slower than kernel expansion (Figures 6–9). This is because kernel detection needs to extract all γ'-quasi-cliques. As shown in Figures 4 and 5, it requires greater computation time to mine all γ'-quasi-cliques when γ' is smaller. On the other hand, if γ' is larger, the γ'-quasi-cliques found by kernel detection are smaller. This fact keeps the size of γ'-quasi-cliques small, at least in the graphs we used. Therefore, since the kernel expansion phase starts with smaller kernels, it requires more time to explore larger γ-quasi-cliques. The ideal choice of γ' should balance between the costs of the two phases, kernel expansion and kernel detection.

B. Impact of the minimum size threshold ($minsize$)

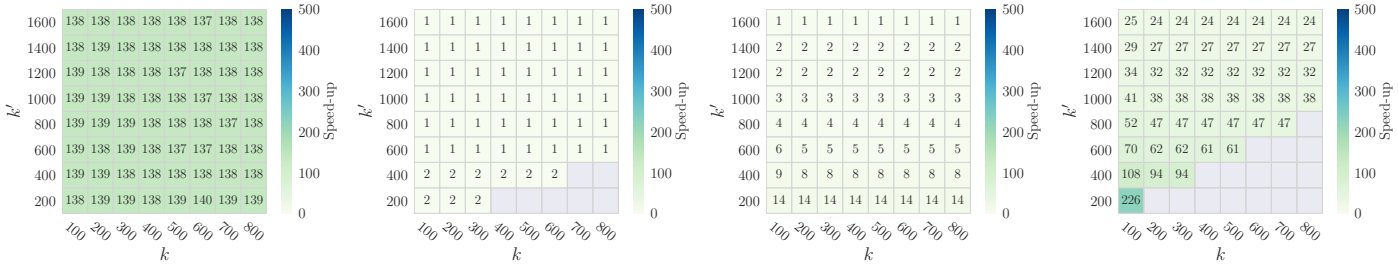
Figure 10 represents the runtimes of KERNELQC and BASELINE for different values of $minsize$. Based on the runtimes of KERNELQC, one can see that $minsize$ does not have a major impact on the runtime, for the most part. However, in some cases, such as Figure 10c and for $minsize = 10$, the runtime of KERNELQC decreases drastically. This is because the size of largest γ-quasi-clique in the graph **Bible** is 12. Setting $minsize = 10$ can considerably reduce the search space for KERNELQC, which leads to a decrease in runtime.

$minsize$ is a user-defined parameter. When there is no knowledge about the given graph and $minsize$ is set to a high value, it is possible that the graph does not contain k γ-quasi-cliques with the size at least $minsize$, let alone top-k maximal γ-quasi-cliques. This can cause us to miss large γ-quasi-cliques which could be good candidates to be placed in top-k maximal γ-quasi-cliques. One way to handle this is to start with a high value of $minsize$ and decrease it if enough quasi-cliques are not found with prior settings.

C. Dependence on k and k'

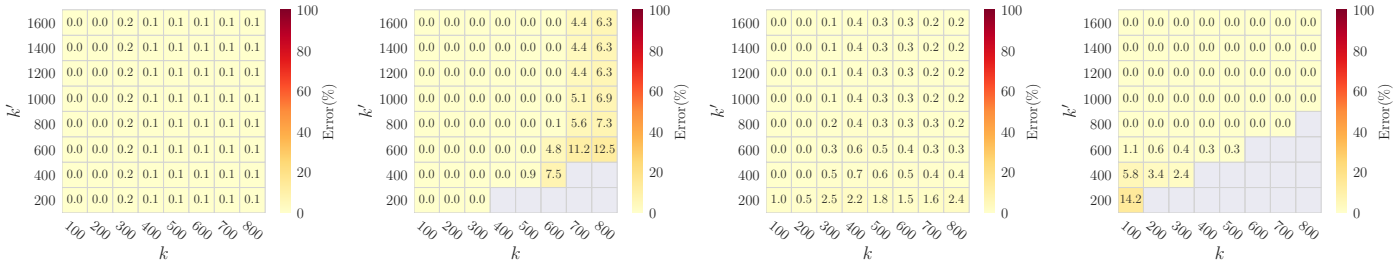
We consider different values of k and k'. Figures 12a–12d show the error percent of KERNELQC. More specifically, each cell shows the error percent for corresponding values of k and k'. In addition, Figures 11a–11d represent the speedup factor of KERNELQC over BASELINE. Similarly, each cell in these figures represent a speedup factor of KERNELQC over BASELINE. There are also some empty cells (for example in Figures 11d and 12d). The empty cells indicate that in some graphs KERNELQC could not extract k maximal γ-quasi-cliques with a given value of k' due to a very few number of maximal quasi-cliques. Therefore, we did not report the error percent and speedup factors for those cases.

Speedup compared to BASELINE: Figures 11a–11d represent the speedup factor of KERNELQC over BASELINE. Based on the results, an increase in value of k' makes KERNELQC slower compared to BASELINE. For example, in Figure 11d, for $k = 100$ and $k' = 200$, the speedup of KERNELQC over BASELINE is 226x while for the same value of k and $k' = 400$, it is reduced to 108x. The reason is that a higher value of k' in KERNELQC means more number of kernels. Therefore, the kernel expansion phase needs to expand more kernels, which increases the overall runtime.



(a) **Advogato** (b) **Route-views** (c) **Bible** (d) **Slash**

Fig. 11: Speedup factor of KERNELQC over BASELINE, for $\gamma = 0.8$, $\gamma' = 1.0$, and $minsize = 3$.



(a) **Advogato** (b) **Route-views** (c) **Bible** (d) **Slash**

Fig. 12: Error percentage of KERNELQC, for $\gamma = 0.8$, $\gamma' = 1.0$, and $minsize = 3$.

Error rate: Here, we describe the effect of parameter k' on the accuracy of KERNELQC. As shown in Figures 12a–12d, the higher value of k' we set, the lower error we obtain. For example, in Figure 12d, for $k = 100$ and $k' = 200$, KERNELQC results in 14.2% error while increasing the value of k' to 800 can yield zero percent error. The reason is that by setting higher values for k', we retrieve more γ'-quasi-cliques in kernel detection of KERNELQC, and there are more kernels to be expanded by the kernel expansion of KERNELQC. In other words, a high value for k' can increase the chance of KERNELQC to unearth very large γ-quasi-cliques. For a fixed value of k', the error percent of different values of k fluctuates slightly in most cases. Here, we give an example why an increase in value of k can results in both lower and higher error percent. Let assume the size of γ-quasi-cliques returned by KERNELQC is $H = \langle 10, 10, 9\rangle$ (for $k = 3$), and the size of γ-quasi-cliques returned by the exact algorithm (BASELINE) is $Z = \langle 12, 10, 10\rangle$. Based on the error metric we used (See Equation (2)), the error percent of KERNELQC in this case is 9.3%. For $k = 4$, suppose that the returned list by BASELINE is $Z = \langle 12, 10, 10, 9\rangle$. The error percent can be lowered if KERNELQC returns $H = \langle 10, 10, 9, 9\rangle$, where the error is 7.3%. It can be also greater if KERNELQC returns $H = \langle 10, 10, 9, 8\rangle$, where the error is 9.7%.

D. Performance of KERNELQC on Large graphs

Our method can handle larger graphs. As shown in Table II, KERNELQC is able to retrieve large maximal quasi-cliques on the graphs with millions of edges and vertices. For example, KERNELQC lists 100 maximal quasi-cliques in 3130 and 9026 secs respectively for the graphs **Youtube** and **Hyves** while BASELINE does not finish after 259K secs (72 hours). The speedup is even higher in the graph **Live-mocha**, where KERNELQC takes only 843 secs while BASELINE did not finish in 72 hours. This is because **Live-mocha** has a higher average degree, hence denser than other graphs (See Table I for more details). Therefore, the search space for BASELINE in this graph can be huge while KERNELQC quickly enumerates kernels of **Live-mocha** and then expands them to obtain k maximal γ-quasi-cliques.

Graph	KERNELQC				BASELINE
	γ	γ'	Avg sz	Time(sec)	Time(sec)
Live-mocha	0.85	1.0	24.1	843	$>$ 259K secs
Youtube	0.8	1.0	27.2	3130	$>$ 259K secs
Hyves	0.75	0.95	33.4	9026	$>$ 259K secs

TABLE II: Performance of KERNELQC on large graphs. $k = 100, k' = 300, minsize = 5$. Avg sz shows the average size of k quasi-cliques. $>$ 259K means BASELINE did not finish in 72 hours (note this happens with every graph).

VI. CONCLUSIONS

Quasi-clique enumeration is an important problem in the area of dense subgraph enumeration. We considered the problem of enumerating top-k maximal degree-based quasi-cliques from a graph. We first showed that it is NP-hard to even determine whether a given (degree-based) quasi-clique is maximal. We then presented a novel heuristic algorithm KERNELQC for enumerating top-k maximal quasi-cliques, based on an idea of finding dense kernels, followed by expanding them into larger quasi-cliques. Our experiments showed that KERNELQC can often lead to a speedup of three orders of magnitude, when compared with a state-of-the-art baseline algorithm. This implies that it may be possible to mine quasi-cliques from larger graphs than was possible earlier. Many directions remain to be explored, including the following: (1) Can the idea of detecting

and expanding kernels be applied to other incomplete dense structures, such as quasi-bicliques? (2) Can the algorithms for quasi-cliques be parallelized effectively?

REFERENCES

[1] C. Bron and J. Kerbosch, "Algorithm 457: finding all cliques of an undirected graph," *Commun. ACM*, vol. 16, no. 9, pp. 575–577, 1973.

[2] E. Tomita, A. Tanaka, and H. Takahashi, "The worst-case time complexity for generating all maximal cliques and computational experiments," *Theoretical Computer Science*, vol. 363, no. 1, pp. 28–42, 2006.

[3] K. Makino and T. Uno, "New algorithms for enumerating all maximal cliques," in *SWAT*, 2004, pp. 260–272.

[4] D. Eppstein, M. Löffler, and D. Strash, "Listing all maximal cliques in sparse graphs in near-optimal time," in *ISAAC*, 2010, pp. 403–414.

[5] A. P. Mukherjee, P. Xu, and S. Tirthapura, "Enumeration of maximal cliques from an uncertain graph," *IEEE Trans. Knowl. Data Eng.*, vol. 29, no. 3, pp. 543–555, 2017.

[6] M. Svendsen, A. P. Mukherjee, and S. Tirthapura, "Mining maximal cliques from a large graph using mapreduce: Tackling highly uneven subproble m sizes," *Journal of Parallel and distributed computing*, vol. 79, pp. 104–114, 2015.

[7] R. D. Alba, "A graph-theoretic definition of a sociometric clique," *Journal of Mathematical Sociology*, vol. 3, pp. 113–126, 1973.

[8] L. C. Freeman, "The sociological concept of "group": An empirical test of two models," *American Journal of Sociology*, vol. 98, no. 1, pp. 152–166, 1992.

[9] V. Spirin and L. A. Mirny, "Protein complexes and functional modules in molecular networks," vol. 100, no. 21, pp. 12 123–12 128, 2003.

[10] M. Bhattacharyya and S. Bandyopadhyay, "Mining the largest quasi-clique in human protein interactome," in *Adaptive and Intelligent Systems, 2009. ICAIS'09. International Conference on*. IEEE, 2009, pp. 194–199.

[11] S. Gunnemann, I. Farber, B. Boden, and T. Seidl, "Subspace clustering meets dense subgraph mining: A synthesis of two paradigms," in *Data Mining (ICDM), 2010 IEEE 10th International Conference on*. IEEE, 2010, pp. 845–850.

[12] B. Boden, S. Günnemann, H. Hoffmann, and T. Seidl, "Mining coherent subgraphs in multi-layer graphs with edge labels," in *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2012, pp. 1258–1266.

[13] A. Silva, W. Meira Jr, and M. J. Zaki, "Mining attribute-structure correlated patterns in large attributed graphs," *Proceedings of the VLDB Endowment*, vol. 5, no. 5, pp. 466–477, 2012.

[14] G. Liu and L. Wong, "Effective pruning techniques for mining quasi-cliques," in *Joint European conference on machine learning and knowledge discovery in databases*. Springer, 2008, pp. 33–49.

[15] G. Pastukhov, A. Veremyev, V. Boginski, and O. A. Prokopyev, "On maximum degree-based-quasi-clique problem: Complexity and exact approaches," *Networks*, vol. 71, no. 2, pp. 136–152, 2018.

[16] H. Matsuda, T. Ishihara, and A. Hashimoto, "Classifying molecular sequences using a linkage graph with their pairwise similarities," *Theoretical Computer Science*, vol. 210, no. 2, pp. 305–325, 1999.

[17] C. Komusiewicz, "Multivariate algorithmics for finding cohesive subnetworks," *Algorithms*, vol. 9, no. 1, p. 21, 2016.

[18] J. Pei, D. Jiang, and A. Zhang, "On mining cross-graph quasi-cliques," in *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*. ACM, 2005, pp. 228–238.

[19] Z. Zeng, J. Wang, L. Zhou, and G. Karypis, "Out-of-core coherent closed quasi-clique mining from large dense graph databases," *ACM Transactions on Database Systems (TODS)*, vol. 32, no. 2, p. 13, 2007.

[20] P. Lee and L. V. Lakshmanan, "Query-driven maximum quasi-clique search," in *Proceedings of the 2016 SIAM International Conference on Data Mining*. SIAM, 2016, pp. 522–530.

[21] J. Abello, M. G. Resende, and S. Sudarsky, "Massive quasi-clique detection," in *Latin American symposium on theoretical informatics*. Springer, 2002, pp. 598–612.

[22] T. Uno, "An efficient algorithm for enumerating pseudo cliques," in *International Symposium on Algorithms and Computation*. Springer, 2007, pp. 402–414.

[23] J. Pattillo, A. Veremyev, S. Butenko, and V. Boginski, "On the maximum quasi-clique problem," *Discrete Applied Mathematics*, vol. 161, no. 1-2, pp. 244–257, 2013.

[24] M. Brunato, H. H. Hoos, and R. Battiti, "On effectively finding maximal quasi-cliques in graphs," in *International conference on learning and intelligent optimization*. Springer, 2007, pp. 41–55.

[25] P. Balister, B. Bollobás, J. Sahasrabudhe, and A. Veremyev, "Dense subgraphs in random graphs," *arXiv preprint arXiv:1803.10349*, 2018.

[26] A. Veremyev, V. Boginski, P. A. Krokhmal, and D. E. Jeffcoat, "Dense percolation in large-scale mean-field random networks is provably explosive," *PloS one*, vol. 7, no. 12, p. e51883, 2012.

[27] N. Chiba and T. Nishizeki, "Arboricity and subgraph listing algorithms," *SIAM J. Comput.*, vol. 14, pp. 210–223, 1985.

[28] A. Conte, R. Grossi, A. Marino, and L. Versari, "Sublinear-space bounded-delay enumeration for massive network analytics: maximal cliques," in *43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016)*, vol. 148, 2016, pp. 1–148.

[29] A. Das, M. Svendsen, and S. Tirthapura, "Change-sensitive algorithms for maintaining maximal cliques in a dynamic graph," *CoRR*, vol. abs/1601.06311, 2016. [Online]. Available: http://arxiv.org/abs/1601.06311

[30] G. Alexe, S. Alexe, Y. Crama, S. Foldes, P. L. Hammer, and B. Simeone, "Consensus algorithms for the generation of all maximal bicliques," *Discrete Applied Mathematics*, vol. 145, no. 1, pp. 11–21, 2004.

[31] G. Liu, K. Sim, and J. Li, "Efficient mining of large maximal bicliques," in *Data warehousing and knowledge discovery*, 2006, pp. 437–448.

[32] A. P. Mukherjee and S. Tirthapura, "Enumerating maximal bicliques from a large graph using mapreduce," *IEEE Trans. Services Computing*, vol. 10, no. 5, pp. 771–784, 2017.

[33] V. Batagelj and M. Zaversnik, "An o (m) algorithm for cores decomposition of networks," *arXiv preprint cs/0310049*, 2003.

[34] J. Cheng, Y. Ke, S. Chu, and M. T. Özsu, "Efficient core decomposition in massive networks," in *Data Engineering (ICDE), 2011 IEEE 27th International Conference on*. IEEE, 2011, pp. 51–62.

[35] N. S. Dasari, R. Desh, and M. Zubair, "Park: An efficient algorithm for k-core decomposition on multicore processors," in *Big Data (Big Data), 2014 IEEE International Conference on*. IEEE, 2014, pp. 9–16.

[36] W. Khaouid, M. Barsky, V. Srinivasan, and A. Thomo, "K-core decomposition of large networks on a single pc," *Proceedings of the VLDB Endowment*, vol. 9, no. 1, pp. 13–23, 2015.

[37] J. Cohen, "Trusses: Cohesive subgraphs for social network analysis," *National Security Agency Technical Report*, vol. 16, 2008.

[38] A. V. Goldberg, *Finding a maximum density subgraph*. University of California Berkeley, CA, 1984.

[39] M. Charikar, "Greedy approximation algorithms for finding dense components in a graph," in *International Workshop on Approximation Algorithms for Combinatorial Optimization*. Springer, 2000, pp. 84–95.

[40] S. Khuller and B. Saha, "On finding dense subgraphs," in *International Colloquium on Automata, Languages, and Programming*. Springer, 2009, pp. 597–608.

[41] R. Andersen and K. Chellapilla, "Finding dense subgraphs with size bounds," in *International Workshop on Algorithms and Models for the Web-Graph*. Springer, 2009, pp. 25–37.

[42] C. E. Tsourakakis, "A novel approach to finding near-cliques: The triangle-densest subgraph problem," *arXiv preprint arXiv:1405.1477*, 2014.

[43] C. Tsourakakis, "The k-clique densest subgraph problem," in *Proceedings of the 24th international conference on world wide web*. International World Wide Web Conferences Steering Committee, 2015, pp. 1122–1132.

[44] M. Mitzenmacher, J. Pachocki, R. Peng, C. Tsourakakis, and S. C. Xu, "Scalable large near-clique detection in large-scale networks via sampling," in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2015, pp. 815–824.

[45] Z. Zou, J. Li, H. Gao, and S. Zhang, "Finding top-k maximal cliques in an uncertain graph," in *Data Engineering (ICDE), 2010 IEEE 26th International Conference on*. IEEE, 2010, pp. 649–652.

[46] O. D. Balalau, F. Bonchi, T. Chan, F. Gullo, and M. Sozio, "Finding subgraphs with maximum total density and limited overlap," in *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining*. ACM, 2015, pp. 379–388.

[47] E. Galbrun, A. Gionis, and N. Tatti, "Top-k overlapping densest subgraphs," *Data Mining and Knowledge Discovery*, vol. 30, no. 5, pp. 1134–1165, 2016.

[48] M. R. Garey and D. S. Johnson, *Computers and intractability*. wh freeman New York, 2002, vol. 29.