Mining Largest Maximal Quasi-Cliques

SEYED-VAHID SANEI-MEHRI, Iowa State University
APURBA DAS, BITS Pilani, Hyderabad Campus
HOOMAN HASHEMI and SRIKANTA TIRTHAPURA, Iowa State University

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 in bioinformatics 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 whether 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, followed 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.

CCS Concepts: • **Computer systems organization** → **Embedded systems**; *Redundancy*; • **Networks** → Network reliability;

Additional Key Words and Phrases: Dense subgraph mining, In-complete subgraphs, Quasi-Clique enumeration, Large-scale graphs, heuristic algorithms

ACM Reference format:
Seyed-Vahid Sanei-Mehri, Apurba Das, Hooman Hashemi, and Srikanta Tirthapura. 2021. Mining Largest Maximal Quasi-Cliques. *ACM Trans. Knowl. Discov. Data* 15, 5, Article 81 (April 2021), 21 pages.
https://doi.org/10.1145/3446637

1 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 [13, 29, 46, 53, 67, 69], which are complete dense structures in a graph. The requirement of complete connectivity among vertices of the graph is often too strict, and edges may be missing among some pairs of vertices, or the existence of some edges

A preliminary version of this work has been published in 2018 IEEE International Conference on Big Data [62].
At the time the article was completed, the author was at Iowa State University.
Authors' addresses: S.-V. Sanei-Mehri, H. Hashemi, and S. Tirthapura, Iowa State University, Ames, Iowa; emails: {vas, hashemi, snt}@iastate.edu; A. Das, BITS Pilani, Hyderabad Campus, Hyderabad, India; email: apurba@hyderabad.bits-pilani.ac.in.

1556-4681/2021/04-ART81 $15.00
https://doi.org/10.1145/3446637

may not be captured during observation. For example, cliques were found to be overly restrictive in identifying cohesive subgroups in social network analysis [2, 30]; instead, dense subgraph models other than cliques were preferred because they relax the complete connectivity requirements. A similar need was found in the analysis of protein–protein interaction networks [66]. This leads to the concept of "incomplete dense structures" or "clique relaxations" that are dense subgraphs where the strict requirement of having edge between every pair of vertices is relaxed. Such concepts lead to more robust methods for identifying dense structures in a graph. In summary, the study of clique relaxations is important in large-scale graph analysis.

In this work, we consider a type of clique relaxation called a *degree-based* quasi-clique in a graph. For a parameter $0 < \gamma \leq 1$, an m-vertex subgraph H of a graph $G = (V, E)$ is a degree-based γ-quasi-clique (henceforth called "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 requires H to be a clique. By increasing γ, it is possible to create 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. 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. Maximality reduces redundancy in the output by ensuring that if a quasi-clique Q is output, no other quasi-clique contained in Q is also output. Note that a maximal quasi-clique may not be the largest (maximum) quasi-clique in the graph.

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 [11], clustering in a multilayer network [12, 34], and exploring correlated patterns from an attributed graph [64]. Here, we list three real-world applications of quasi-clique mining in detail.

- **Criminal Activity Detection.** Criminals are unlikely to reveal their personal data in social networks; therefore, identifying them through profile information is not a practical approach. However, their interactions with other criminals on the underlying network can help to uncover criminal groups. Studies such as [7, 41, 57] show that the 19 hijackers of the tragic terrorist attack of September 11, 2001 form a tightly knit subgraph that resembles a quasi-clique (or is interpreted as a clique with missing edges) in the underlying network, constructed by Krebs [41]. Note that the data used to construct the network were publicly available even before the tragic attack. Therefore, mining tightly connected subgraphs and further processing and gathering information about the user accounts involved in the dense subgraphs could have uncovered the terrorist group.
- **Neurological Disorder Detection.** Neurological disorders are caused by an abnormality of the structure and function of the central nervous system or peripheral nervous system. Genetic and environmental factors are the root of some diseases such as Alzheimer's and Parkinson's. According to Centers for Disease Control and Prevention, approximately 60,000 Americans are diagnosed with Parkinson's disease each year; in 2017 alone, 32,964 Americans died of this disease, which is identified as the second most common neurological disorder in the United States [49]. In a brain network, the correlation network for Parkinson's disease is identified by mining densely-connected regions in protein–protein interaction networks, which uncovers significant pathways, such as the Parkinson's disease pathway [68]. This process, i.e., analyzing the structure and function of brain networks using dense subgraph mining algorithms, is critical to understanding the causes and mechanisms of disease progression and, thus, promoting better treatments and aiding in drug discovery for Parkinson's disease.

— **Applications in Biological Networks.** Biological networks contain a molecular landscape that is basically a biological process in living cells. In a biological network, many proteins interact with each other to form larger molecular machines that perform complex molecular functions. A protein complex is indeed formed by a group of proteins which are tightly connected through strong and long-lasting interactions [45]. In other words, protein complexes are interpreted as dense subgraphs in the underlying network and are considered to be one of the most important types of molecular machines. Extracting dense subgraphs (such as quasi-cliques) in a biological network is, therefore, one of the best-known computational tasks in biology which helps biologists to explore the structural and functional properties of the underlying biological network.

We consider top-k maximal quasi-clique enumeration, where it is required to enumerate the k largest maximal quasi-cliques in a graph. 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 many as $\Omega(3^{n/3})$ maximal cliques in a graph [51]; hence, there can be at least as many maximal quasi-cliques, as 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 less than the time required for enumerating all maximal quasi-cliques.

A straightforward approach for enumerating top-k maximal quasi-cliques is to first enumerate all maximal quasi-cliques in a graph using an existing algorithm for quasi-clique enumeration such as Quick [44], which is a state-of-the-art exact algorithm for degree-based quasi-clique enumeration, followed by extracting the k largest among them. Certainly, this approach is not efficient from the computation time perspective. When the number of maximal quasi-cliques is much larger than k, most of the enumerated quasi-cliques are discarded and the resulting computation is wasteful. We address this challenge by designing a heuristic algorithm motivated by an empirical observation that a large quasi-clique contains smaller but denser quasi-cliques within it. To the best of our knowledge, we are the first to propose a heuristic algorithm for mining top-k degree-based maximal quasi-cliques. Our heuristic algorithm is orders of magnitude faster than the only competitor (i.e., Quick), and it keeps intact the quality of the solution with respect to the exact algorithm in most cases. We summarize our contributions as follows:

NP-hardness of maximality: At a high level, checking for the maximality of a quasi-clique is not easy because this structure does not follow the hereditary property, meaning that a subgraph of a quasi-clique may not be a quasi-clique. We prove the strong result that the problem of detecting whether a given quasi-clique in a graph is maximal is an NP-hard problem. This is unlike the case of cliques that follow the hereditary property—detecting the maximality of a clique can be done in polynomial time, through simply checking whether it is possible to add one more vertex to the clique. Note that our result is not about checking maximum-sized quasi-cliques [55], which is known to be an NP-complete problem. Here, we show that checking for maximality of a quasi-clique is NP-hard.

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 to enumerate γ-quasi-cliques. Further, the resulting set of γ'-quasi-cliques can be expanded into γ-quasi-cliques more easily than constructing the set of γ-quasi-cliques 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**,[1] KernelQC yields a nearly 1,000-fold speedup for enumerating the top 100 0.7-quasi-cliques as compared to a baseline based on the Quick algorithm [44].

While KernelQC is not guaranteed to return exactly the set of top-k maximal quasi-cliques, we show, through an empirical evaluation, that the accuracy is high in practice. Note that the high cost of enumeration follows, given that the problem of even checking the maximality of a quasi-clique is NP-hard. In many cases that we considered, the output of KernelQC exactly matched the output of the exact algorithm that used exhaustive search. Usually, the distance of the output, when compared to the output of the exact algorithm, was less than 10^{-4}. See section 5 for more details on the metrics used to measure the distance and performance of KernelQC over the baseline algorithm. *Significantly, KernelQC was able to scale to much larger graphs than current methods.*

1.1 Related Works

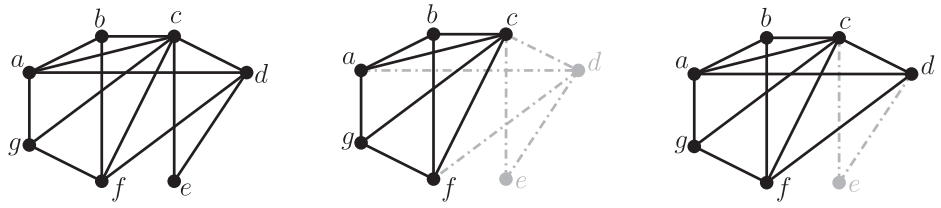
Degree-based and density-based quasi-cliques: The problem of finding γ-quasi-cliques appears in a wide spectrum of applications such as biological [44, 58], social [71, 75], telecommunication [59], and financial [65]. Motivated by a study on protein sequences, Matsuda et al. [48] first defined the degree-based 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 [40]. Pei et al. [58] studied the problem of enumerating those degree-based γ-quasi-cliques from a graph database that occurred in every graph of the graph database. It was the first study on quasi-clique search problem. Jiang and Pei [37] and Zeng et al. [77] studied the same problem as Pei et al. [58] but generalized in the sense that their algorithm enumerated 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. Although [77] and [58] proposed algorithms for finding complete sets of quasi-cliques in a graph, they did not exploit pruning techniques. Liu and Wong [44] proposed the algorithm Quick 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. Recently, Guo et al. [35] developed parallel algorithms, powered by load balancing techniques, to enumerate maximal quasi-cliques in a distributed framework. Lee and Lakshmanan [42] investigated the problem of finding a maximum γ-quasi-clique containing a given subset of vertices S of the original graph and proposed a heuristic algorithm. Pastukhov et al. [55] studied the maximum degree-based γ-quasi-clique problem. First, they proved that finding a maximum γ-quasi-clique is an NP-hard problem and presented algorithms for a γ-quasi-clique of maximum cardinality. It is worth noting that while this work focused on finding a single quasi-clique of the largest size, our goal is not simply to find a single large quasi-clique; rather, it is to enumerate the k largest maximal quasi-cliques. Further, the NP-hardness result in [55] was for finding the maximum γ-quasi-clique, while our NP-hardness result is for finding out whether a quasi-clique is maximal.

Abello et al. [1] first studied 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

[1]The details of the graphs used in the experiments are presented in Section 5.

in a complete subgraph of size $|Q|$ being at least δ. Note that a degree-based quasi-clique is also a density-based quasi-clique, but the inverse is not true. They proposed a heuristic algorithm for finding a large δ-quasi-clique. Haraguchi and Okubo [36] designed a method for finding useful clusters of web pages using quasi-cliques. To obtain a cluster of similar web pages, they exploited Singular Value Decomposition in a term-document matrix created by a corpus to find semantic correlations. Thereafter, similarities among web pages were calculated based on those correlations to make clusters. They represented the set of web pages in a graph, and clusters were found as quasi-cliques in it. The experimental results in this work showed that their method can perform well in finding valuable clusters in a graph. Following this work, Uno [73] considered density-based quasi-cliques and proposed an algorithm for enumerating all δ-quasi-cliques, with polynomial delay. In another study, Pattillo et al. [56] proved that deciding whether there exists a δ-quasi-clique of size at least θ is an NP-complete problem. Brunato et al. [14] defined 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 proposed a heuristic algorithm for finding a maximum (γ, δ)-quasi-clique. Recently, Balister et al. [8] further investigated the work of Veremyev et al. [74], in which an upper bound on the largest γ-quasi-clique is presented. Balister et al. studied the largest order of γ-quasi-clique and derived the concentration bound on the size of the maximum density-based δ-quasi-clique. Other exact and heuristic methods for maximum density-based quasi-cliques include [28, 47, 60].

Other works on dense subgraphs: The study of dense subgraphs has been of high interest in research community for many decades. There have been numerous works on graphlet mining [19, 54, 63], on complete dense subgraphs such as maximal cliques [13, 18, 23–26, 29, 46, 53, 69], maximal bicliques [3, 43, 52], and graph summarization based on maximal cliques and quasi-cliques [76]. There exist many different types of degree-based incomplete dense subgraphs other than the quasi-clique, such as k-core [9, 17, 27, 38], k-truss [20], and k-plex [10, 21, 22]. A k-core is a maximal connected subgraph such that each vertex in that subgraph has a degree of at least k; this subgraph is quite different from a 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 k-truss is an absolute threshold. k-plex is an induced subgraph, where induced degree of a vertex is at least $n-k$ with n being the size of the k-plex. For $n(1-\gamma)+\gamma \leq k < n(1-\gamma)+\gamma+1$, a k-plex with size n is a γ-quasi-clique; however, in general, we cannot conclude that any k-plex is a γ-quasi-clique or vice versa because we do not know n. Indeed, k-plexes are different from quasi-cliques because the number of (allowable) missing edges of a vertex in a quasi-clique is a function of n while the number of missing edges of each vertex of a k-plex is fixed and up to $k-1$. Other works on dense subgraphs different from the quasi-clique subgraph include the densest subgraph [4, 15, 33, 39], triangle densest subgraph [72], k-clique densest subgraph [50, 70], and dense subgraphs in dynamic networks whose edges appear in short time intervals [61]. 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. [78] 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. [6] and Galbrun et al. [31] on the enumeration of top-k densest subgraphs. Another variant of the densest subgraph problem is Tsourakakis et al.'s work [71] in which the notion of the quasi-clique is used to define an alternative of density. Angel et al. [5] studied the set of all subgraphs exceeding a density threshold where the weights of edges are altered under streaming conditions. Chen and Saad [16]



(a) Input Graph G (b) Quasi-Clique (c) Maximal Quasi-Clique

Fig. 1. (a) G is the input graph. (b) Vertices $\{a, b, c, f, g\}$ form a γ-quasi-clique with $\gamma = 0.6$ and *minsize* = 5. (c) Vertices $\{a, b, c, d, f, g\}$ form a *maximal* γ-quasi-clique with $\gamma = 0.6$ and *minsize* = 5.

proposed a matrix-blocking model to identify dense subgraphs. In their proposed method, they were searching for those subgraphs which best cover the input graph.

2 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 2.1 (γ-quasi-clique). For parameter $0 < \gamma \leq 1$ and integer *minsize*, a vertex-induced subgraph Q of G is called a γ-quasi-clique if Q is connected, $|Q| \geq$ *minsize*, 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 another γ-quasi-clique Q' does not exist, such that $V(Q) \subset V(Q')$. See Figure 1 for an example of the above definition.

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

Here, we briefly describe algorithm Quick due to Liu and Wong [44] because we utilized a modified version of this algorithm in our proposed method for enumerating top-k maximal quasi-cliques.

Quick algorithm: Quick is the state-of-the-art algorithm for QCE [44] that enumerates all maximal quasi-cliques using a depth first search-based technique. While this enumeration is prohibitively expensive, due to the exponential number of quasi-cliques (e.g., for $\gamma = 1$, quasi-cliques are indeed cliques, and the number of maximal cliques can be exponential [51]), a widely used technique called pruning can significantly reduce the search space by removing the unpromising candidate vertices. Quick performs several pruning techniques considering factors such as the degree of the vertices and the bounds on the number of vertices that can be added to a growing quasi-clique. The pruning techniques are described in detail in [44]. At each step, Quick adds an eligible vertex to the existing quasi-clique, removes the vertices from the candidate set that are deemed ineligible considering the pruning rules, and expands the quasi-clique as much as possible by calling the main procedure recursively. Quick also applies a post-processing step to filter

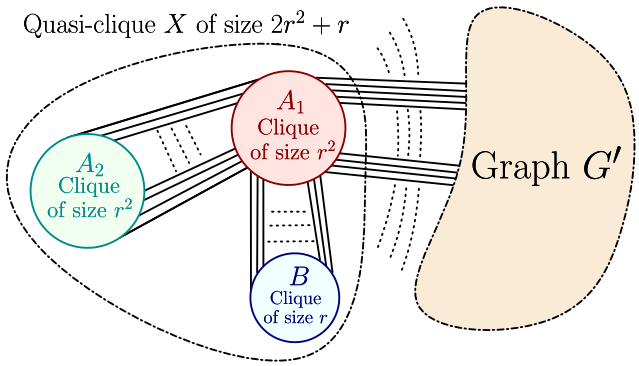


Fig. 2. The construction of graph G for the proof of Theorem 2.

out any enumerated quasi-clique that is contained within other quasi-cliques. Indeed, Quick removes all non-maximal quasi-cliques using the post-processing step, which is time-consuming. We develop QuickM, a variant of Quick, which skips the post-processing step to obtain a better runtime while our heuristic method still reports only maximal quasi-cliques. We present QuickM in Section 4.

3 HARDNESS OF CHECKING THE MAXIMALITY OF A QUASI-CLIQUE

It is easy to deduce that γ-QCE is an NP-hard problem because 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 whether a given quasi-clique is maximal is an NP-hard problem. This is very different from the case of maximal cliques—checking whether a given clique is maximal can be done in polynomial time, by simply checking whether there exists a vertex outside the clique that is connected to all vertices within the clique. If such a vertex exists, then the given clique is not maximal; otherwise, it is maximal.

Problem 2 (Maximality of a Quasi-Clique). *Given a graph $G = (V, E)$ and a γ-quasi-clique $X \subseteq V$, determine whether 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, which asks whether a given graph $G' = (V', E')$ contains a clique of size r, to the problem of checking the maximality of a quasi-clique. This r-clique problem is NP-complete [32]. Given graph G' on which we must 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 $2r^2 + r$ 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:

- —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 $\lceil \gamma \cdot (2r^2 + r - 1) \rceil = \lceil \frac{(r^2+r-1)(2r^2+r-1)}{2r^2+2r-1} \rceil \leq (r^2+r-1)$. 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. Because $X \cap L = \emptyset$, we have $|Q| = |X| + |L| = 2r^2 + 2r$. Therefore, $\lceil \gamma \cdot (|Q| - 1) \rceil = \lceil \frac{(r^2+r-1)(2r^2+2r-1)}{2r^2+2r-1} \rceil = r^2 + r - 1$. 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 were the case, then the minimum degree threshold for a vertex in R would be $\lceil \gamma \cdot (|R| - 1) \rceil = \lceil \frac{(r^2+r-1)(|R|-1)}{2r^2+2r-1} \rceil = \lceil \frac{(r^2+r-1)(2r^2+r+|M|-1)}{2r^2+2r-1} \rceil > r^2 + r - 1$, as $|M| > r$. 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's assume that this was the case. Then, the minimum degree threshold for a vertex in R would be $\lceil \gamma \cdot (|R| - 1) \rceil = \lceil \gamma \cdot (2r^2 + r + |M| - 1) \rceil$. However, the minimum degree of a vertex in R is $r^2 + |M| - 1$ (consider a vertex from M). It can be verified that because $|M| < r$, $\lceil \gamma \cdot (2r^2 + r + |M| - 1) \rceil > r^2 + |M| - 1$. 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 at most $r^2 + r - 2$ (consider a vertex from M). However, the minimum degree threshold for a vertex in R is $\lceil \gamma \cdot (|R| - 1) \rceil = \lceil \frac{(r^2+r-1)(2r^2+2r-1)}{2r^2+2r-1} \rceil = r^2 + r - 1 > r^2 + r - 2$. This completes the proof. □

4 ALGORITHM FOR TOP-k QCE

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

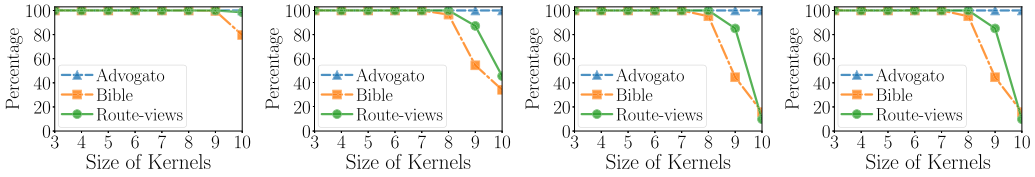
4.1 Baseline

A straightforward algorithm for Top-k QCE is to enumerate all maximal γ-quasi-cliques using the Quick algorithm [44] and to then extract the k largest among them. We call this algorithm Baseline. While this algorithm is simple and generates an exact solution to our problem, it is computationally expensive because the total number of maximal quasi-cliques can be exponential.[2] On the other hand, we showed that the maximality checking of a quasi-clique is an NP-hard problem. Therefore, developing an exact and efficient solution to enumerate maximal quasi-cliques is quite challenging. To end this, we developed KernelQC, a heuristic algorithm that generates fewer quasi-cliques through the appropriate choice of parameter γ which runs orders of magnitude faster than Baseline with high accuracy, such that in most cases the output of our method matches the output of the exact solution. We present the results in Section 5.

4.2 KernelQC Algorithm

We now present our heuristic algorithm, KernelQC, for Top-k QCE. The idea of KernelQC is built upon our empirical observations which can be stated as follows: Within a larger γ-quasi-clique, there are denser and relatively large γ'-quasi-cliques, where $\gamma' > \gamma$. We call these denser quasi-cliques kernels of a γ-quasi-clique. That being said, our algorithm KernelQC can discover a γ-quasi-clique by first extracting its kernels with degree-threshold γ' and then expanding the

[2]The number of maximal cliques in a graph can be as many as $\Omega(3^{n/3})$ [51]; hence, there can be at least as many maximal quasi-cliques, when $\gamma = 1$.



(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 top 1,000 maximal 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**. The average sizes of top 1,000 maximal 0.8-quasi-cliques for **Advogato**, **Route-views**, and **Bible** are 23.49, 12.24, and 12.37, respectively.

kernels with parameter γ. Experiments on several graphs show that our two-step heuristic method extracts large γ-quasi-cliques and is orders of magnitude faster than the naïve method which mines γ-quasi-cliques directly from scratch.

Here, we briefly show results to verify the effectiveness of KernelQC. We considered graphs **Advogato**, **Route-views**, and **Bible**[3], and $\gamma = 0.8$. We looked into the 1,000 largest maximal γ-quasi-cliques each from graphs **Advogato**, **Route-views**, and **Bible**.[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 with a size of at least 7, for different values of γ', ranging from 0.85 to 1.0.* Note that the average sizes of top 1,000 maximal γ-quasi-cliques for graphs **Advogato**, **Route-views**, and **Bible** are 23.49, 12.24, and 12.37, respectively, which suggests that large γ-quasi-cliques (usually) contain γ'-quasi-cliques of substantial sizes as subgraphs for $\gamma' > \gamma$. Details are shown in Figure 3.

Note that an adversary can form a γ-quasi-clique without any large γ'-quasi-clique contained within. This can happen either if the density of a quasi-clique is not relatively high, i.e., the number of edges is relatively low compared to the total number of possible edges, or if the difference between hyperparameters γ and γ' is high. For example, within a $(4, 4)$-biclique which is a 0.5-quasi-clique, there is no kernel with $\gamma' = 0.7$ (assuming $minsize \geq 3$). 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 illustrate 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 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 steps:

(1) *Kernel Detection:* Find kernels in the graph, i.e., γ'-quasi-cliques for the chosen value of γ'. Then, among all kernels, the 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 [44], to enumerate all maximal quasi-cliques that contain each kernels. Next, among all extracted γ-quasi-cliques, the largest-k maximal γ-quasi-cliques are enumerated.

[3]These graphs are described in Table 1.
[4]We did not consider graph **Slash** because the the size of largest γ-quasi-clique in this graph is less than 10.

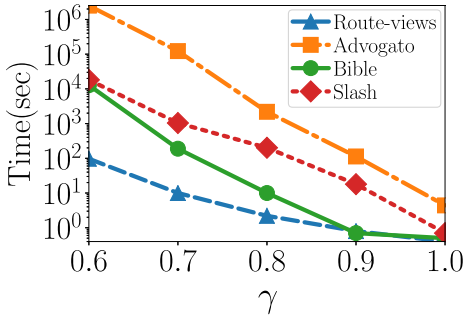


Fig. 4. The runtimes of γ-quasi-clique enumeration for different values of γ, for *minsize* = 5, and for different graphs.

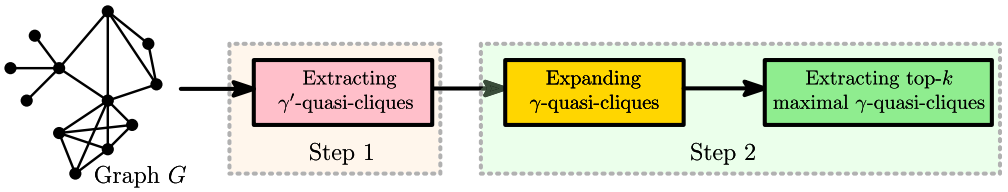


Fig. 5. A schematic of algorithm KernelQC.

Figure 5 briefly sketches two steps of algorithm KernelQC. Algorithm 1 describes KernelQC in detail. 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 whether a quasi-clique is maximal, before outputting it. For KernelQC, the quasi-cliques from the subroutine do not need to be maximal, as Algorithm 2 sorts quasi-cliques in the descending order of their sizes and suppresses non-maximal quasi-cliques. By omitting a maximality check, QuickM is more efficient than Quick. 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 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, because all γ-quasi-cliques in S are sorted in the descending 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. □

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 graph G. Then, in line 3 and by Lemma 1, X contains the largest maximal γ'-quasi-cliques of G, where $|X| \leq k'$. In lines 5 and 6, every γ-quasi-clique of set Z contains at least a γ'-quasi-clique from set Y. This is because we expand every γ'-quasi-clique in Y by QuickM using the parameter γ. Therefore, any quasi-clique in set Z has at least *minsize* vertices. Because $R \subseteq Z$ (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

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 by 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.

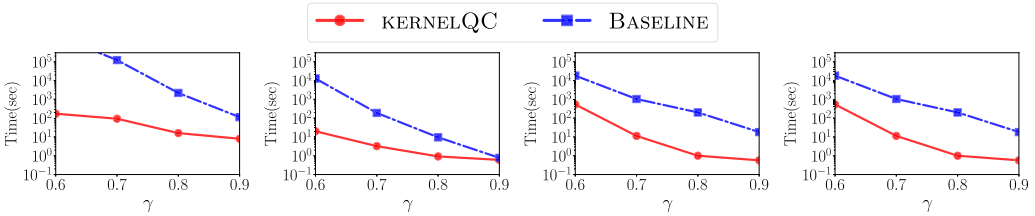
1 Sort S in the descending order of the sizes of quasi-cliques
2 $Q \leftarrow \emptyset$
3 **for** *a quasi-clique* $q \in S$ **do**
4 $\quad$ **if** $(|Q| < k) \wedge (\forall q' \in Q, q \not\subseteq q')$ **then**
5 $\quad\quad Q \leftarrow Q \cup q$
6 **return** Q

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, because $h \subset h'$, 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 must choose two user-defined parameters γ' and k', based on the given values of γ and k. Here, we discuss the impact of these parameters on the accuracy and runtime of KernelQC.

Dependence on γ': For a given γ, varying the value $\gamma' \in (\gamma, 1]$ 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 are relatively smaller in sizes, and the expansion phase will take longer. 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 are relatively larger in sizes; hence, they have less chance to be expanded.

Dependence on k': In Algorithm 1, k' determines the number of kernels, which must be extracted in the kernel detection and then expanded to mine γ-quasi-cliques. The higher the value of k', the greater the number of kernels required 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 because more kernels must 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 5, we present an empirical sensitivity analysis of parameters *minsize*, γ, γ', k, and k' on the accuracy and runtime of the algorithm.



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

Fig. 6. The runtimes of KernelQC and Baseline as a function of γ.

Table 1. The Summary of Input Graphs

Graph	#Vertices	#Edges	Max Degree	Avg Degree
Advogato	5,155	39,285	803	15.24
Route-views	6,474	12,572	1,458	3.88
Bible	1,773	9,131	364	10.30
Slash	51,083	116,573	2,915	4.56
Live-mocha	104,103	2,193,083	2,980	42.13
Youtube	1,134,890	2,987,624	28,754	5.27
Hyves	1,402,673	2,777,419	31,883	3.96

5 EXPERIMENTS

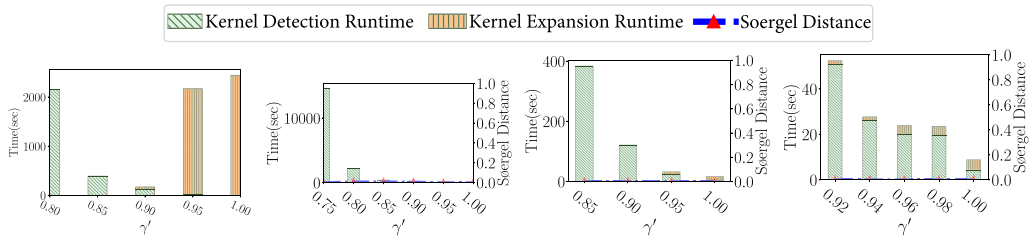
Networks and experimental setup: We used real-world networks from a network repository at KONECT. The networks are summarized in Table 1 and are converted to simple graphs by removing self-loops and multiple edges. We implemented the Baseline and KernelQC algorithms in C++[5] and compiled them with g++ compiler with -O3 as the optimization level. The experiments are conducted on a cluster of machines equipped with 2.0 GHz 8-Core Intel E5 2650 and 64.0 GB memory.

Metrics: As discussed in Section 4, 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 distance, which is as follows. Suppose that $H = \langle h_1, h_2, h_3, \ldots, h_k \rangle$ is a descending ordered list, maintaining the sizes of k maximal γ-quasi-cliques returned by KernelQC.[6] Similarly, suppose that $Z = \langle z_1, z_2, z_3, \ldots, z_k \rangle$ is a list in descending order, which contains the sizes of the top-k maximal γ-quasi-cliques, returned by Baseline, the exact algorithm. The søergel distance between the two lists H and Z is as follows:

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

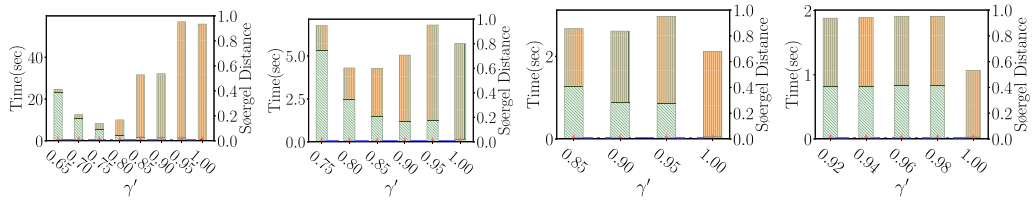
For our purpose, we used søergel distance (also known as tanimoto distance) because the output lists contain non-negative numbers, representing the sizes of largest maximal quasi-cliques. Also, this metric compares two lists element-wise, which fits our need, as i^{th} element in a list shows the size of i^{th} largest maximal quasi-clique. The søergel distance is better suited than other metrics

[5] https://github.com/beginner1010/topk-quasi-clique-enumeration.
[6] A size of a quasi-clique is the number of vertices in the quasi-clique.



(a) $\gamma = 0.6$ did not finish after 259K secs. (b) $\gamma = 0.7$ Baseline takes 122K secs. (c) $\gamma = 0.8$ Baseline takes 2185 secs. (d) $\gamma = 0.9$ Baseline takes 113 secs.

Fig. 7. The accuracy and runtime of KernelQC on graph **Advogato**, with $k = 100$, $k' = 300$, *minsize* $= 5$. The søergel distance is 0 in (b), (c), and (d).



(a) $\gamma = 0.6$ Baseline takes 97 secs. (b) $\gamma = 0.7$ Baseline takes 10 secs. (c) $\gamma = 0.8$ Baseline takes 2 secs. (d) $\gamma = 0.9$ Baseline takes 0.9 secs.

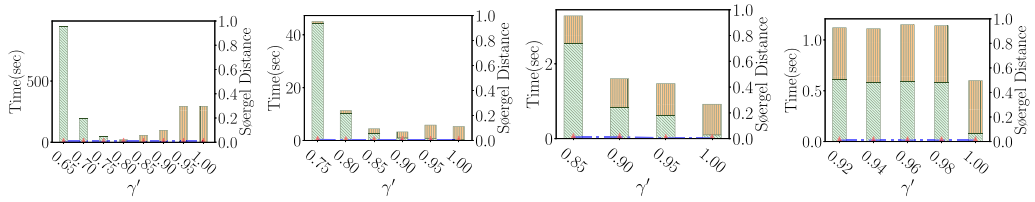
Fig. 8. The accuracy and runtime of KernelQC on graph **Route-views**, with $k = 100$, $k' = 300$, *minsize* $= 5$. The søergel distance is 0 for the four values of γ.

such as Jaccard similarity. In particular, if we used Jaccard similarity to measure the similarity between the 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 size, but whose elements are different, the Jaccard similarity will show a poor match, while the søergel distance will show a perfect match. Note that the søergel distance shows the similarity of two lists of *non-negative* numbers. Here, we consider the lists of *sizes* of quasi-cliques, obtained by KernelQC and Baseline algorithms. As mentioned above, the two lists are of length k and are sorted in order of descending size of quasi-cliques. Henceforth, when we refer to the output distance of KernelQC, we compare the returned lists of KernelQC and Baseline using Equation (1).

Runtime compared to Baseline: The experiments show that KernelQC yields a significant speedup over Baseline for enumerating Top-k-quasi-cliques. For example, in Figure 7(b) in graph **Advogato** with $\gamma = 0.7$, $k = 100$, and $k' = 300$, when we set $\gamma' = 0.9$, KernelQC yields a speedup of 984× over Baseline. For $\gamma = 0.6$, the speedup is even more because Baseline did not finish after 259K seconds, while KernelQC took only 172 seconds.[7] For graph **Bible**, with $\gamma = 0.6$ and $\gamma' = 0.8$, KernelQC yields a 34× speedup over Baseline (Figure 9(a)). In **Slash** and the same values for γ and γ', KernelQC yields a 638× speedup over Baseline (Figure 10(a)). 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 Baseline had a small runtime (< 100 seconds). For this graph, obtaining high speedup is not as important.

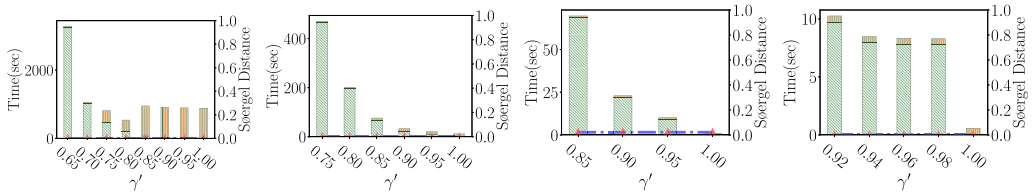
Søergel distance: The results show that KernelQC has high accuracy for different graphs and various parameter settings while achieving a huge speedup over Baseline. As shown in

[7]K stands for thousands.



(a) $\gamma = 0.6$ BASELINE takes 12K secs. (b) $\gamma = 0.7$ BASELINE takes 189 secs. (c) $\gamma = 0.8$ BASELINE takes 10 secs. (d) $\gamma = 0.9$ BASELINE takes 0.8 secs.

Fig. 9. The accuracy and runtime of KERNELQC on graph **Bible**, with $k = 100$, $k' = 300$, $minsize = 5$, The maximum soergel distance is 0, 0.001, 0.008, and 0.006 for (a), (b), (c), and (d), respectively.



(a) $\gamma = 0.6$ BASELINE takes 18152 secs. (b) $\gamma = 0.7$ BASELINE takes 1039 secs. (c) $\gamma = 0.8$ BASELINE takes 201 secs. (d) $\gamma = 0.9$ BASELINE takes 18 secs.

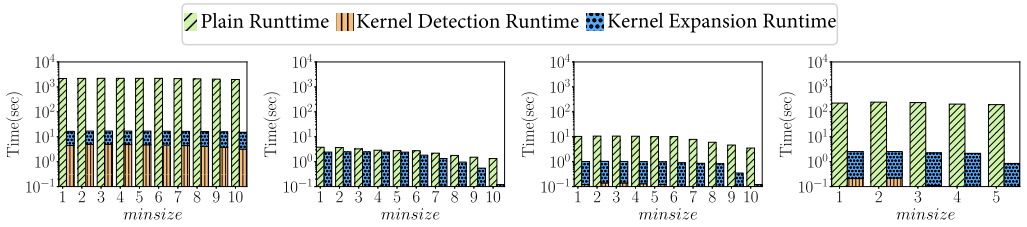
Fig. 10. The accuracy and runtime of KERNELQC on graph **Slash**, with $k = 100$, $k' = 300$, $minsize = 5$. The maximum soergel distance for (c) is 0.02 and for (a), (b), and (d) is 0.

Figures 7–10, the soergel distance for most graphs and different values of γ and γ' is less than 9×10^{-3} and is often zero (i.e., exactly matches the output of BASELINE). The highest distance measure among all experiments is 2.1×10^{-2} and belongs to the graph **Slash**, where γ is 0.8 (Figure 10(c)). We did not report the distance measure of Figure 7(a) because BASELINE did not finish after 259K seconds.

5.1 Dependence on γ and γ'

In Figure 7–10, 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 for choosing an optimum value for γ' which can result in a good balance between accuracy and runtime. For all graphs, we set $k = 100$, $k' = 300$ to find top-k maximal 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 impact the performance of the two steps of KERNELQC. When γ' is close to γ, kernel detection is slower than kernel expansion (Figure 7—10). This is because kernel detection must extract all γ'-quasi-cliques. As shown in Figures 4 and 6, greater computation time is required 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, because 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 computational costs of the two phases, i.e., kernel expansion and kernel detection.



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

Fig. 11. The runtimes of KernelQC and Baseline as a function of *minsize*, for $\gamma = 0.8$, $\gamma' = 1.0$, $k = 100$, and $k' = 300$.

5.2 Impact of the Minimum Size Threshold (*minsize*)

Figure 11 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 11(c) 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 lower 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 a size of at least *minsize*, let alone top-k maximal γ-quasi-cliques. This can cause missing large γ-quasi-cliques which could be good candidates for being 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.

5.3 Dependence on k and k'

We consider different values of k and k'. Figures 13(a)–13(d) show the søergel distance of KernelQC. More specifically, each cell shows the søergel distance for corresponding values of k and k'. In addition, Figures 12(a)–12(d) represent the speedup factor of KernelQC over Baseline. Similarly, each cell in these figures represents a speedup factor of KernelQC over Baseline. There are also some empty cells (for example, in Figures 12(d) and 13(d)). 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 low number of maximal quasi-cliques. Therefore, we did not report the distance metric and speedup factors for those cases.

Speedup compared to Baseline: Figures 12(a)–12(d) represent the speedup factor of KernelQC over Baseline. Based on the results, an increase in the value of k' makes KernelQC slower compared to Baseline. For example, in Figure 12(d), for $k = 100$ and $k' = 200$, the speedup of KernelQC over Baseline is 226× while for the same value of k and $k' = 400$, it is reduced to 108×. The reason is that a higher value of k' in KernelQC means a higher number of kernels. Therefore, the kernel expansion phase must expand more kernels, which increases the overall runtime.

Søergel distance: Here, we describe the effect of parameter k' on the accuracy of KernelQC. As shown in Figures 13(a)–13(d), the higher the value of k' we set, the lower the output distance we obtain. For example, in Figure 13(d), for $k = 100$ and $k' = 200$, the søergel distance of KernelQC is 0.1 while increasing the value of k' to 800 can yield zero distance. The reason is that by setting higher values for k', we retrieve more γ'-quasi-cliques in the 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 that KernelQC will unearth very large γ-quasi-cliques. For a fixed value of k', the søergel distance of different values of k fluctuates slightly in most cases.

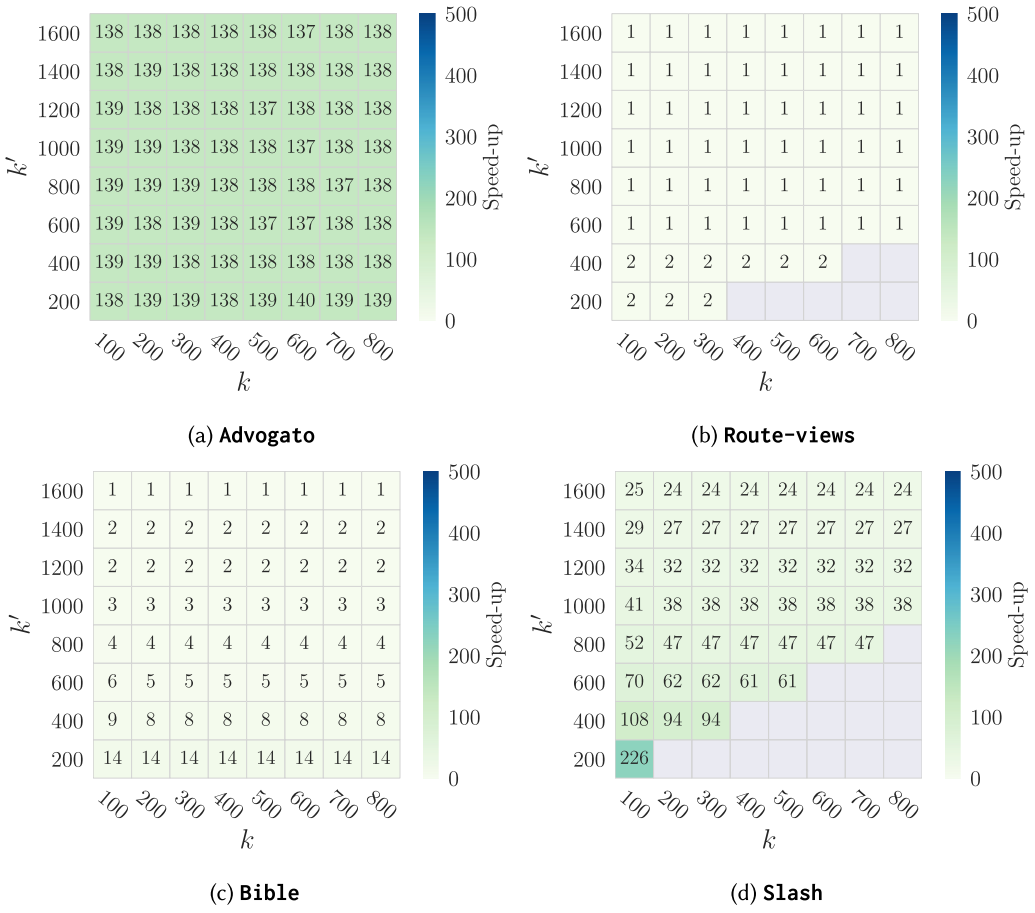
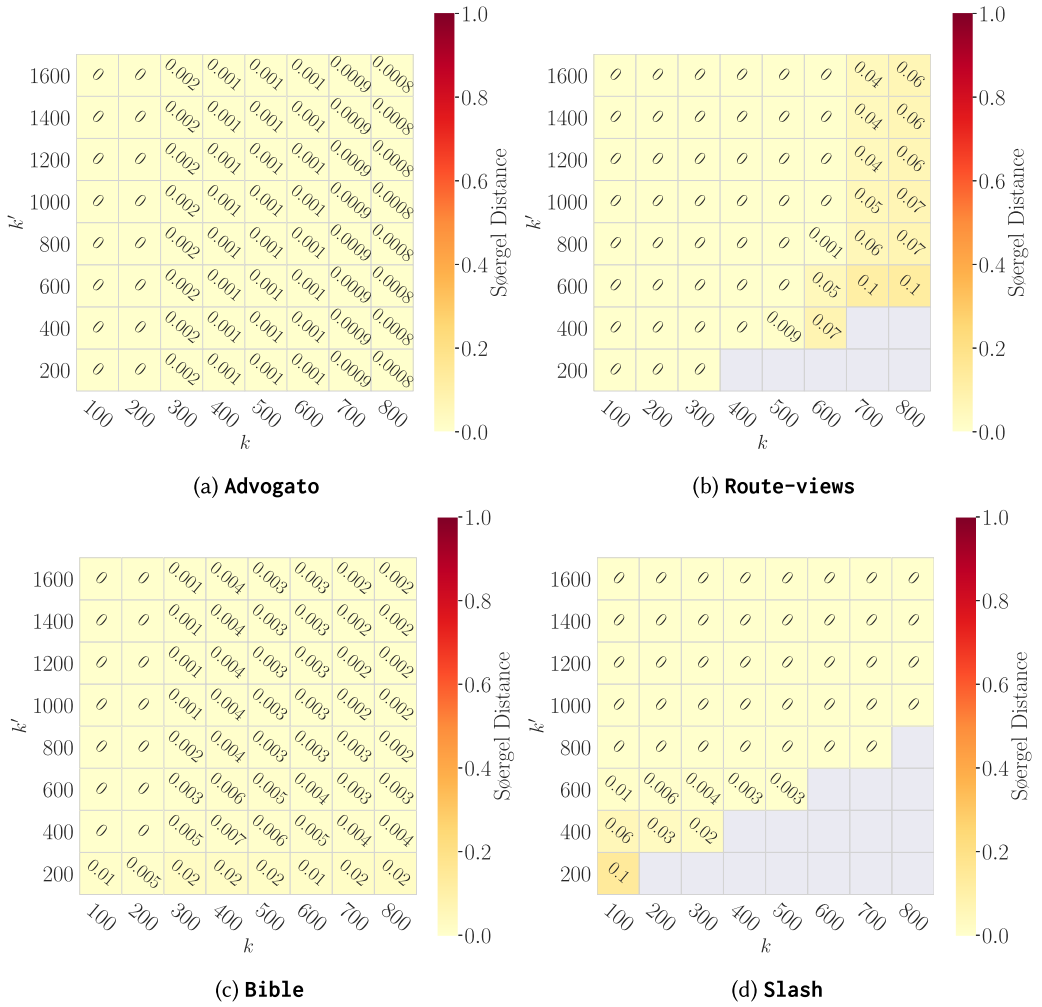


Fig. 12. The speedup factor of KernelQC over Baseline, for $\gamma = 0.8$, $\gamma' = 1.0$, and *minsize* = 3.

Here, we provide an example of why an increase in the value of k can result in both lower and higher søergel distances. Let's assume that 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 distance metric we used (see Equation (1)), the output distance of KernelQC in this case is 0.093. For $k = 4$, suppose that the list returned by Baseline is $Z = \langle 12, 10, 10, 9\rangle$. The søergel distance is lowered if KernelQC returns $H = \langle 10, 10, 9, 9\rangle$, where the distance is 0.073. The distance is higher if KernelQC returns $H = \langle 10, 10, 9, 8\rangle$, where the distance is 0.097.

5.4 The Performance of KernelQC on Large Graphs

Our method can handle larger graphs. As shown in Table 2, KernelQC can retrieve large maximal quasi-cliques on the graphs with millions of edges and vertices. For example, KernelQC lists 100 maximal quasi-cliques in 3,130 and 9,026 seconds, respectively, for the graphs **Youtube** and **Hyves** while Baseline does not finish after 259K seconds (72 hours). The speedup is even higher in the graph **Live-mocha**, where KernelQC takes only 843 seconds while Baseline did not finish in 72 hours. This is because **Live-mocha** has a higher average degree and, hence, is denser than other graphs (see Table 1 for more details). Therefore, the search space for Baseline in this graph can



(a) **Advogato** (b) **Route-views**

(c) **Bible** (d) **Slash**

Fig. 13. The accuracy of KernelQC, for $\gamma = 0.8$, $\gamma' = 1.0$, and *minsize* = 3.

be huge while KernelQC quickly enumerates kernels of **Live-mocha** and then expands them to obtain k maximal γ-quasi-cliques.

6 CONCLUSION

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 often leads to a speedup of three orders of magnitude as compared to a state-of-the-art baseline algorithm, while the output of KernelQC matches the output of the exact solution in most cases. 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 and k-plexes? (2)

Table 2. The Performance of KernelQC on Large Graphs

			KernelQC		Baseline
Graph	γ	γ'	Avg sz	Time(sec)	Time(sec)
Live-mocha	0.85	1.0	24.1	**843**	>259K
Youtube	0.8	1.0	27.2	**3,130**	>259K
Hyves	0.75	0.95	33.4	**9,026**	>259K

With parameters $k = 100$, $k' = 300$, $minsize = 5$. Avg sz shows the average size of k quasi-cliques, found by KernelQC. >259K indicates that Baseline did not finish in 72 hours (note this occurs with every graph).

Can the algorithms for quasi-cliques be parallelized effectively? Generally speaking, one major challenge in proposing an efficient parallel solution in dense subgraph mining is to obtain high load balancing among sub-tasks [25]. For example, a parallel solution in a shared-memory setting must consider the balanced workload of sub-divided tasks among threads to achieve a considerable speedup, which is non-trivial for mining maximal quasi-cliques because the computational workload highly depends on the structure of the graph for each sub-task. (3) Can KernelQC be modified to reduce overlaps between the maximal quasi-cliques? Note that in the current version of KernelQC, there might be overlaps among the extracted maximal quasi-cliques. However, our method KernelQC is pliable enough to embrace other heuristic methods for reducing overlaps among reported maximal quasi-cliques. For example, one can devise a heuristic algorithm and apply it to the expansion of kernels, or one can modify algorithm QuickM to lower the overlaps among quasi-cliques in both kernel detection and expansion steps.

REFERENCES

[1] James Abello, Mauricio G.C. Resende, and Sandra Sudarsky. 2002. Massive quasi-clique detection. In *Proceedings of the 5th Latin American Symposium on Theoretical Informatics*. Springer, 598–612.

[2] Richard D. Alba. 1973. A graph-theoretic definition of a sociometric clique. *Journal of Mathematical Sociology* 3 (1973), 113–126.

[3] Gabriela Alexe, Sorin Alexe, Yves Crama, Stephan Foldes, Peter L. Hammer, and Bruno Simeone. 2004. Consensus algorithms for the generation of all maximal bicliques. *Discrete Applied Mathematics* 145, 1 (2004), 11–21.

[4] Reid Andersen and Kumar Chellapilla. 2009. Finding dense subgraphs with size bounds. In *Proceedings of the International Workshop on Algorithms and Models for the Web-Graph*. Springer, 25–37.

[5] Albert Angel, Nikos Sarkas, Nick Koudas, and Divesh Srivastava. 2012. Dense subgraph maintenance under streaming edge weight updates for real-time story identification. *Proceedings of the VLDB Endowment* 5, 6 (2012), 574–585.

[6] Oana Denisa Balalau, Francesco Bonchi, T.H. Chan, Francesco Gullo, and Mauro Sozio. 2015. Finding subgraphs with maximum total density and limited overlap. In *Proceedings of the 8th ACM International Conference on Web Search and Data Mining*. ACM, 379–388.

[7] Balabhaskar Balasundaram, Sergiy Butenko, and Illya V. Hicks. 2011. Clique relaxations in social network analysis: The maximum k-plex problem. *Operations Research* 59, 1 (2011), 133–142.

[8] Paul Balister, Béla Bollobás, Julian Sahasrabudhe, and Alexander Veremyev. 2019. Dense subgraphs in random graphs. *Discrete Applied Mathematics* 260 (2019), 66–74.

[9] Vladimir Batagelj and Matjaz Zaversnik. 2003. An O (m) algorithm for cores decomposition of networks. *arXiv preprint cs/0310049* (2003).

[10] Devora Berlowitz, Sara Cohen, and Benny Kimelfeld. 2015. Efficient enumeration of maximal k-plexes. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*. 431–444.

[11] Malay Bhattacharyya and Sanghamitra Bandyopadhyay. 2009. Mining the largest quasi-clique in human protein interactome. In *Proceedings of the International Conference on Adaptive and Intelligent Systems (ICAIS'09)*. IEEE, 194–199.

[12] Brigitte Boden, Stephan Günnemann, Holger Hoffmann, and Thomas Seidl. 2012. 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, 1258–1266.

[13] Coen Bron and Joep Kerbosch. 1973. Algorithm 457: Finding all cliques of an undirected graph. *Communications of the ACM* 16, 9 (1973), 575–577.
[14] Mauro Brunato, Holger H. Hoos, and Roberto Battiti. 2007. On effectively finding maximal quasi-cliques in graphs. In *Proceedings of the International Conference on Learning and Intelligent Optimization*. Springer, 41–55.
[15] Moses Charikar. 2000. Greedy approximation algorithms for finding dense components in a graph. In *Proceedings of the International Workshop on Approximation Algorithms for Combinatorial Optimization*. Springer, 84–95.
[16] Jie Chen and Yousef Saad. 2012. Dense subgraph extraction with application to community detection. *IEEE Transactions on Knowledge and Data Engineering* 24, 7 (2012), 1216–1230.
[17] James Cheng, Yiping Ke, Shumo Chu, and M. Tamer Özsu. 2011. Efficient core decomposition in massive networks. In *Proceedings of the 2011 IEEE 27th International Conference on Data Engineering (ICDE'11)*. IEEE, 51–62.
[18] Norishige Chiba and Takao Nishizeki. 1985. Arboricity and subgraph listing algorithms. *SIAM Journal on Computing* 14, 1 (1985), 210–223.
[19] Shumo Chu and James Cheng. 2012. Triangle listing in massive networks. *ACM Transactions on Knowledge Discovery from Data* 6, 4 (2012), 17.
[20] Jonathan Cohen. 2008. Trusses: Cohesive subgraphs for social network analysis. *National Security Agency Technical Report* 16.
[21] Alessio Conte, Tiziano De Matteis, Daniele De Sensi, Roberto Grossi, Andrea Marino, and Luca Versari. 2018. D2K: Scalable community detection in massive networks via small-diameter k-plexes. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1272–1281.
[22] Alessio Conte, Donatella Firmani, Caterina Mordente, Maurizio Patrignani, and Riccardo Torlone. 2017. Fast enumeration of large k-plexes. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 115–124.
[23] Alessio Conte, Roberto Grossi, Andrea Marino, and Luca Versari. 2016. Sublinear-space bounded-delay enumeration for massive network analytics: Maximal cliques. In *Proceedings of the 43rd International Colloquium on Automata, Languages, and Programming (ICALP'16)*, Vol. 148. 1–148.
[24] Apurba Das, Seyed-Vahid Sanei-Mehri, and Srikanta Tirthapura. 2018. Shared-memory parallel maximal clique enumeration. In *Proceedings of the 2018 IEEE 25th International Conference on High Performance Computing (HiPC'18)*. IEEE, 62–71.
[25] Apurba Das, Seyed-Vahid Sanei-Mehri, and Srikanta Tirthapura. 2020. Shared-memory parallel maximal clique enumeration from static and dynamic graphs. *ACM Transactions on Parallel Computing* 7, 1 (2020), 1–28.
[26] Apurba Das, Michael Svendsen, and Srikanta Tirthapura. 2016. Change-sensitive algorithms for maintaining maximal cliques in a dynamic graph. *CoRR* abs/1601.06311 (2016). http://arxiv.org/abs/1601.06311
[27] Naga Shailaja Dasari, Ranjan Desh, and Mohammad Zubair. 2014. ParK: An efficient algorithm for k-core decomposition on multicore processors. In *Proceedings of the 2014 IEEE International Conference on Big Data (Big Data'14)*. IEEE, 9–16.
[28] Youcef Djeddi, Hacene Ait Haddadene, and Nabil Belacel. 2019. An extension of adaptive multi-start tabu search for the maximum quasi-clique problem. *Computers & Industrial Engineering* 132 (2019), 280–292.
[29] David Eppstein, Maarten Löffler, and Darren Strash. 2010. Listing all maximal cliques in sparse graphs in near-optimal time. In *Proceedings of the International Symposium on Algorithms and Computation*. 403–414.
[30] Linton C. Freeman. 1992. The sociological concept of "Group": An empirical test of two models. *American Journal of Sociology* 98, 1 (1992), 152–166.
[31] Esther Galbrun, Aristides Gionis, and Nikolaj Tatti. 2016. Top-k overlapping densest subgraphs. *Data Mining and Knowledge Discovery* 30, 5 (2016), 1134–1165.
[32] Michael R. Garey and David S. Johnson. 2002. *Computers and Intractability*. Vol. 29. W.H. Freeman, New York.
[33] Andrew V. Goldberg. 1984. *Finding a Maximum Density Subgraph*. University of California Berkeley, CA.
[34] Stephan Gunnemann, Ines Farber, Brigitte Boden, and Thomas Seidl. 2010. Subspace clustering meets dense subgraph mining: A synthesis of two paradigms. In *Proceedings of the 2010 IEEE 10th International Conference on Data Mining (ICDM'10)*. IEEE, 845–850.
[35] Guimu Guo, Da Yan, M. Tamer Özsu, and Zhe Jiang. 2020. Scalable mining of maximal quasi-cliques: An algorithm-system codesign approach. *arXiv preprint arXiv:2005.00081* (2020).
[36] Makoto Haraguchi and Yoshiaki Okubo. 2006. A method for pinpoint clustering of web pages with pseudo-clique search. In *Federation Over the Web*. Springer, 59–78.
[37] Daxin Jiang and Jian Pei. 2009. Mining frequent cross-graph quasi-cliques. *ACM Transactions on Knowledge Discovery from Data* 2, 4 (2009), 16.
[38] Wissam Khaouid, Marina Barsky, Venkatesh Srinivasan, and Alex Thomo. 2015. K-core decomposition of large networks on a single PC. *Proceedings of the VLDB Endowment* 9, 1 (2015), 13–23.

[39] Samir Khuller and Barna Saha. 2009. On finding dense subgraphs. In *Proceedings of the 36th International Colloquium on Automata, Languages, and Programming*. Springer, 597–608.
[40] Christian Komusiewicz. 2016. Multivariate algorithmics for finding cohesive subnetworks. *Algorithms* 9, 1 (2016), 21.
[41] Valdis E. Krebs. 2002. Mapping networks of terrorist cells. *Connections* 24, 3 (2002), 43–52.
[42] Pei Lee and Laks V.S. Lakshmanan. 2016. Query-driven maximum quasi-clique search. In *Proceedings of the 2016 SIAM International Conference on Data Mining*. SIAM, 522–530.
[43] Guimei Liu, Kelvin Sim, and Jinyan Li. 2006. Efficient mining of large maximal bicliques. In *Data Warehousing and Knowledge Discovery*. 437–448.
[44] Guimei Liu and Limsoon Wong. 2008. Effective pruning techniques for mining quasi-cliques. In *Proceedings of the Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 33–49.
[45] Xiuli Ma, Guangyu Zhou, Jingbo Shang, Jingjing Wang, Jian Peng, and Jiawei Han. 2017. Detection of complexes in biological networks through diversified dense subgraph mining. *Journal of Computational Biology* 24, 9 (2017), 923–941.
[46] Kazuhisa Makino and Takeaki Uno. 2004. New algorithms for enumerating all maximal cliques. In *Proceedings of the Scandinavian Workshop on Algorithm Theory*. 260–272.
[47] Fabrizio Marinelli, Andrea Pizzuti, and Fabrizio Rossi. 2018. A star-based reformulation for the maximum quasi-clique problem. In *Proceedings of the 16th Cologne-Twente Workshop on Graphs and Combinatorial Optimization*. 112.
[48] Hideo Matsuda, Tatsuya Ishihara, and Akihiro Hashimoto. 1999. Classifying molecular sequences using a linkage graph with their pairwise similarities. *Theoretical Computer Science* 210, 2 (1999), 305–325.
[49] Diane B. Miller and James P. O'Callaghan. 2015. Biomarkers of Parkinson's disease: Present and future. *Metabolism* 64, 3 (2015), S40–S46.
[50] Michael Mitzenmacher, Jakub Pachocki, Richard Peng, Charalampos Tsourakakis, and Shen Chen Xu. 2015. 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, 815–824.
[51] John W. Moon and Leo Moser. 1965. On cliques in graphs. *Israel Journal of Mathematics* 3, 1 (1965), 23–28.
[52] Arko Provo Mukherjee and Srikanta Tirthapura. 2017. Enumerating maximal bicliques from a large graph using mapreduce. *IEEE Transactions on Services Computing* 10, 5 (2017), 771–784.
[53] Arko Provo Mukherjee, Pan Xu, and Srikanta Tirthapura. 2017. Enumeration of maximal cliques from an uncertain graph. *IEEE Transactions on Knowledge and Data Engineering* 29, 3 (2017), 543–555.
[54] Ha-Myung Park, Francesco Silvestri, Rasmus Pagh, Chin-Wan Chung, Sung-Hyon Myaeng, and U. Kang. 2018. Enumerating trillion subgraphs on distributed systems. *ACM Transactions on Knowledge Discovery from Data* 12, 6 (2018), 71.
[55] Grigory Pastukhov, Alexander Veremyev, Vladimir Boginski, and Oleg A. Prokopyev. 2018. On maximum degree-based-quasi-clique problem: Complexity and exact approaches. *Networks* 71, 2 (2018), 136–152.
[56] Jeffrey Pattillo, Alexander Veremyev, Sergiy Butenko, and Vladimir Boginski. 2013. On the maximum quasi-clique problem. *Discrete Applied Mathematics* 161, 1–2 (2013), 244–257.
[57] Jeffrey Pattillo, Nataly Youssef, and Sergiy Butenko. 2012. Clique relaxation models in social network analysis. In *Handbook of Optimization in Complex Networks*. Springer, 143–162.
[58] Jian Pei, Daxin Jiang, and Aidong Zhang. 2005. On mining cross-graph quasi-cliques. In *Proceedings of the 11th ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*. ACM, 228–238.
[59] Sergio Rajsbaum. 2002. *LATIN 2002: Theoretical Informatics: 5th Latin American Symposium, Cancun, Mexico, April 3-6, 2002, Proceedings*. Vol. 5. Springer Science & Business Media.
[60] Celso C. Ribeiro and José A. Riveaux. 2019. An exact algorithm for the maximum quasi-clique problem. *International Transactions in Operational Research* 26, 6 (2019), 2199–2229.
[61] Polina Rozenshtein, Nikolaj Tatti, and Aristides Gionis. 2017. Finding dynamic dense subgraphs. *ACM Transactions on Knowledge Discovery from Data* 11, 3 (2017), 27.
[62] Seyed-Vahid Sanei-Mehri, Apurba Das, and Srikanta Tirthapura. 2018. Enumerating top-k quasi-cliques. In *Proceedings of the 2018 IEEE International Conference on Big Data (Big Data'18)*. IEEE, 1107–1112.
[63] Seyed-Vahid Sanei-Mehri, Ahmet Erdem Sariyuce, and Srikanta Tirthapura. 2018. Butterfly counting in bipartite networks. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 2150–2159.
[64] Arlei Silva, Wagner Meira Jr., and Mohammed J. Zaki. 2012. Mining attribute-structure correlated patterns in large attributed graphs. *Proceedings of the VLDB Endowment* 5, 5 (2012), 466–477.
[65] Kelvin Sim, Jinyan Li, Vivekanand Gopalkrishnan, and Guimei Liu. 2006. Mining maximal quasi-bicliques to co-cluster stocks and financial ratios for value investment. In *Proceedings of the 6th International Conference on Data Mining (ICDM'06)*. IEEE, 1059–1063.
[66] Victor Spirin and Leonid A. Mirny. 2003. Protein complexes and functional modules in molecular networks. 100, 21 (2003), 12123–12128. DOI: https://doi.org/10.1073/pnas.2032324100

[67] Michael Svendsen, Arko Provo Mukherjee, and Srikanta Tirthapura. 2015. Mining maximal cliques from a large graph using mapreduce: Tackling highly uneven subproble m sizes. *Journal of Parallel and Distributed Computing* 79 (2015), 104–114.

[68] Jaya Thomas, Dongmin Seo, and Lee Sael. 2016. Review on graph clustering and subgraph similarity based analysis of neurological disorders. *International Journal of Molecular Sciences* 17, 6 (2016), 862.

[69] Etsuji Tomita, Akira Tanaka, and Haruhisa Takahashi. 2006. The worst-case time complexity for generating all maximal cliques and computational experiments. *Theoretical Computer Science* 363, 1 (2006), 28–42.

[70] Charalampos Tsourakakis. 2015. The k-clique densest subgraph problem. In *Proceedings of the 24th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 1122–1132.

[71] Charalampos Tsourakakis, Francesco Bonchi, Aristides Gionis, Francesco Gullo, and Maria Tsiarli. 2013. Denser than the densest subgraph: Extracting optimal quasi-cliques with quality guarantees. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 104–112.

[72] Charalampos E. Tsourakakis. 2014. A novel approach to finding near-cliques: The triangle-densest subgraph problem. *arXiv preprint arXiv:1405.1477* (2014).

[73] Takeaki Uno. 2007. An efficient algorithm for enumerating pseudo cliques. In *Proceedings of the International Symposium on Algorithms and Computation*. Springer, 402–414.

[74] Alexander Veremyev, Vladimir Boginski, Pavlo A. Krokhmal, and David E. Jeffcoat. 2012. Dense percolation in large-scale mean-field random networks is provably "explosive". *PloS One* 7, 12 (2012), e51883.

[75] Chrysafis Vogiatzis, Alexander Veremyev, Eduardo L. Pasiliao, and Panos M. Pardalos. 2015. An integer programming approach for finding the most and the least central cliques. *Optimization Letters* 9, 4 (2015), 615–633.

[76] Ling Wang, Yu Lu, Bo Jiang, Kai Tai Gao, and Tie Hua Zhou. 2020. Dense subgraphs summarization: An efficient way to summarize large scale graphs by super nodes. In *Proceedings of the International Conference on Intelligent Computing*. Springer, 520–530.

[77] Zhiping Zeng, Jianyong Wang, Lizhu Zhou, and George Karypis. 2007. Out-of-core coherent closed quasi-clique mining from large dense graph databases. *ACM Transactions on Database Systems* 32, 2 (2007), 13.

[78] Zhaonian Zou, Jianzhong Li, Hong Gao, and Shuo Zhang. 2010. Finding top-k maximal cliques in an uncertain graph. In *Proceedings of the 2010 IEEE 26th International Conference on Data Engineering (ICDE'10)*. IEEE, 649–652.

Received April 2020; revised November 2020; accepted December 2020