Quasi-Clique Mining for Graph Summarization

Antoine Castillon[1,2], Julien Baste[1], Hamida Seba[2], and Mohammed Haddad[2](✉)

[1] Univ. Lille, CNRS, Centrale Lille, UMR 9189 CRIStAL, 59000 Lille, France
{antoine.castillon,julien.baste}@univ-lille.fr
[2] Univ Lyon, UCBL, CNRS, INSA Lyon, LIRIS, UMR5205, 69622 Villeurbanne, France
{hamida.seba,mohammed.haddad}@univ-lyon1.fr

Abstract. Several graph summarization approaches aggregate dense sub-graphs into super-nodes leading to a compact summary of the input graph. The main issue for these approaches is how to achieve a high compression rate while retaining as much information as possible about the original graph structure within the summary. These approaches necessarily involve an algorithm to mine dense structures in the graph such as quasi-clique enumeration algorithms. In this paper, we focus on improving these mining algorithms for the specific task of graph summarization. We first introduce a new pre-processing technique to speed up this mining step. Then, we extend existing quasi-clique enumeration algorithms with this filtering technique and apply them to graph summarization.

Keywords: Graph summarization · Quasi-clique mining · Pruning techniques

1 Background

Nowadays, several large-scale systems such as social networks or the link structure of the World Wide Web involve graphs with millions and even billions of nodes and edges [3]. The analysis of such graphs can prove to be very difficult given the huge amount of information contained in them. The goal of *graph summarization* (or *graph compression*) is to provide, given an input graph, a smaller summary [5]. Depending of the situation, this summary can assume different roles. Either it exists uniquely to save storage space and can be decompressed to obtain the original graph with or without loss of information, or it can be used directly to obtain information on the input graph such as paths, neighbourhoods and clusters. When summarizing a graph, one can focus on the optimisation of the size of the summary introducing a size threshold [1,5] or an information loss threshold [8]. Others adapt the summary so as to optimise some kind of queries [1,4].

This work is supported by Agence Nationale de la Recherche (ANR-20-CE23-0002).

Even if these summarization methods perform well in term of execution time and compression rate, they do not retain all the information available in the input graph, especially the dense components which are mainly *cliques*, and *quasi-cliques* which are essential in the analysis of many real life networks. This motivated several graph summarization approaches based on dense subgraph mining [8,10]. These methods aggregate dense subgraphs into supernodes and keep track of all non-edges inside the supernode as correcting edges, to ensure lossless compression and allow the reconstruction of the original graph.

In this paper, we focus on enhancing the enumeration of dense subgraphs, and more precisely quasi-cliques, for the purpose of graph summarization. We first introduce a new pruning technique which allows an important speed up of quasi-clique enumeration algorithms. Then, we describe how to adapt the search for quasi-cliques to graph summarization by solving best suited relaxations of the Quasi-Clique Enumeration Problem.

The remainder of this document is organised as follows. Section 2 gives formal definitions of the concepts used in the paper. Section 3 introduces the new pruning technique and presents a relaxation of the quasi-clique enumeration problem. Section 4 compares the performances of the different quasi-clique mining schemes and their application to dense subgraph based summarization.

2 Preliminaries

In this paper all graphs are simple, undirected and consist of two sets: the vertex set V and the edge set E formed by pairs of V. Given $G = (V, E)$ a clique of G is a subset C of V such that the vertices in C are pairwise adjacent. We use the following definition for quasi-cliques which are a generalization of cliques to other dense subgraphs.

Definition 1 (γ-quasi-clique). *Given $G = (V, E)$, and $\gamma \in [0, 1]$, a subset of vertices $Q \subseteq V$ is called a γ-quasi-clique if for all $v \in V$, $d_Q(v) \geq \gamma(|Q| - 1)$, where $N_Q(v)$ represents the neighbours of v in Q and $d_Q(v) = |N_Q(v)|$.*

Definition 2 (Quasi-Clique Enumeration Problem). *Given $G = (V, E)$, a density threshold $\gamma \in [0, 1]$ and a size threshold m, the* Quasi-Clique Enumeration Problem (QCE) *consists of finding all maximal γ-quasi-cliques in G of size greater than m.*

QCE is well known to be difficult (the associated decision problem being NP-hard [2]), and have already been studied a lot trough several aspects. We focus on the exhaustive enumeration and its most classic method: the Quick algorithm [7] which performs a depth-first exploration of the solution space tree and it prunes the branches which cannot lead to a new solution. During the search, the algorithm keeps in memory X the current subset and a set `cand` formed by vertices outside X which are the next candidates to add to X. The pruning occurs according to these sets and several rules, and actually corresponds to the removal of vertices from `cand`. These rules are mostly based on the diameter and the degree threshold of quasi-cliques. For example, with $\gamma > 0.5$ the diameter of a γ-quasi-clique is at most 2.

3 Contributions

3.1 A New Pruning Technique

While the Quick algorithm is efficient in the general case, it is not hard to point out cases where it is not. One can, for instance, take the example depicted in Fig. 1. The pruning rules of the Quick algorithm are based only on the vertices and their degree, hence, in the case where each vertex is contained in a quasi-clique, the pruning techniques cannot be properly applied and the algorithm proceeds to the exploration of almost the whole solution space tree.

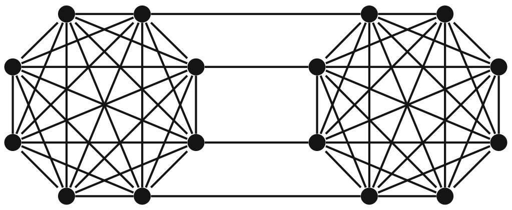

Fig. 1. Example of the inefficiency of the Quick algorithm.

In order to handle these cases, we introduce a new technique based on sets of vertices. According to Definition 1, each vertex v in a quasi-clique Y has at least $\lceil\gamma(|Y|-1)\rceil$ neighbours in Y. This property motivates the following pruning rule (see Algorithm 1): given the density and size threshold γ and m, if $\{u, v\} \in E$ is such that u and v share strictly less than $2\lceil\gamma(m-1)\rceil - m$ common neighbours then we can remove $\{u, v\}$ from E.

Algorithm 1. Pre_processing(G, γ, m)

```
repeat
    delete all $u \in V$ s.t. $d(u) < \lceil\gamma(m-1)\rceil$
    delete all $\{u, v\} \in E$ s.t. $|N(u) \cap N(v)| < 2\lceil\gamma(m-1)\rceil - m$
until $G$ has not been modified during the last loop
```

3.2 A Redundancy-Free Graph Summarization Approach

While the Quick algorithm is an effective method to tackle the QCE problem, it is important to note that this problem is not perfectly suited for graph summarization. Indeed, having access to all quasi-cliques can create redundancy in the summary and impact the performances. To deal with this issue, we adapt the concept of visibility, introduced in [9] for cliques, to detect and avoid redundant quasi-cliques. The visibility of a maximal quasi-clique Q with respect to a set of maximal quasi-cliques $\mathcal{S}$ is a value between 0 and 1 which describes how well Q

is represented by the quasi-cliques of $\mathcal{S}$. If the visibility is 1 then Q is in $\mathcal{S}$, on the contrary if the visibility is 0 then Q does not overlap with any quasi-clique of $\mathcal{S}$.

Definition 3 (Visibility). *Given a graph G, $\gamma \in [0,1]$, $Q \in \mathcal{M}_\gamma(G)$, the set of all maximal γ quasi-cliques of G, and $\mathcal{S} \subseteq \mathcal{M}_\gamma(G)$, the* ***$\mathcal{S}$-visibility*** *of Q: $\mathcal{V}_\mathcal{S}(Q)$, is defined as:*

$$\mathcal{V}_\mathcal{S}(Q) = \max_{Q' \in \mathcal{S}} \frac{|Q \cap Q'|}{|Q|}.$$

This leads to a variant of the QCE problem where instead of finding all the quasi-cliques one only have to ensure that all maximal quasi-cliques have a sufficient visibility. We propose to use a redundancy-aware quasi-clique mining algorithm to summarize graphs. This will give direct access to the dense subgraphs in the summary while avoiding redundant quasi-cliques. This will improve not only compression rates but also allows to speed up the computation.

To implement this approach, we rely on the following observation. Given a graph G and two quasi-cliques Q, Q'. If Q and Q' are very similar ($\mathcal{V}_{\{Q\}}(Q')$ is close to 1) then most of the edges in $G[Q']$ are already covered in $G[Q]$. On the contrary if Q and Q' are very different then $G[Q]$ and $G[Q']$ do not share many edges. Hence, to avoid redundancy, after finding the quasi-clique Q one can remove from G all the edges inside $G[Q]$, thus the quasi-cliques of G similar to Q become sparse and are skipped by the pruning techniques of the Quick algorithm. Removing such edges usually does not impact other different quasi-cliques since these quasi-cliques do not share many edges with Q. This method depicted in Algorithm 2 outputs a set of quasi-cliques $\mathcal{S}$ which covers all edges contained in quasi-cliques.

Algorithm 2. Redundancy_aware(G, γ, m)

```
run Quick($G, \gamma, m$)
    each time a quasi-clique $X$ is found:
        remove from $G$ all edges of $G[X]$
```

4 Performances

We compare the performances of three different quasi-clique mining algorithms: the Quick algorithm presented in Sect. 1, Algorithm 2 presented in Sect. 3.2 and finally a greedy quasi-clique mining algorithm, which serves as a basis of comparison. We implemented the three algorithms in Python 3 using a Windows 10 machine with a 2.5 GHz Intel(R) Core (TM) i5 processor and 8 GB RAM. We tested each algorithm with and without our pre-processing technique, namely Algorithm 1, on several types of graphs. We present, here, our results on real graphs from social networks such as Facebook (with 4039 vertices and 88,234 edges), Twitter (with 81,306 vertices and 1,768,149 edges) and Google+ (with 107,614 vertices and 13,673,453 edges), from the Stanford large network dataset collection [6]. We measure not only the runtime of the algorithms but also how well they are suited to graph summarization. For these experiments, we vary the number of nodes in the considered graphs from 0 to 1750 by growing subgraphs from the considered graphs. The obtained results are the average on several experiments.

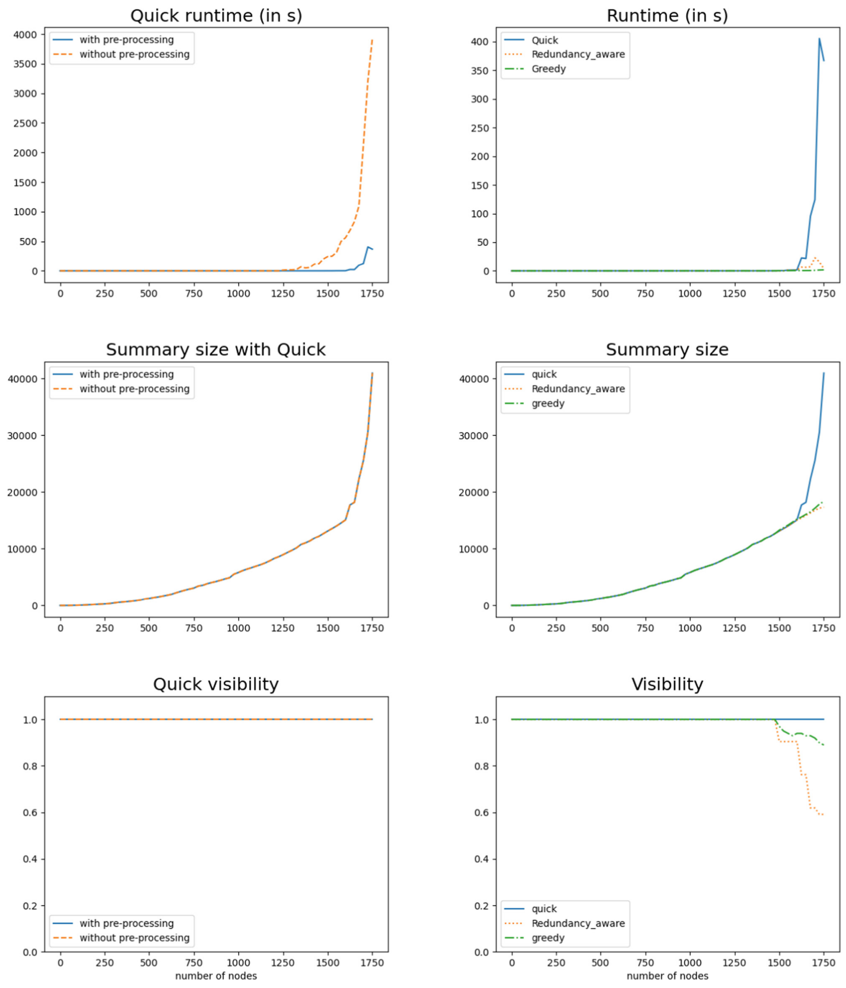

Fig. 2. Performances of the algorithms on social networks.

Figure 2 presents our results. The left side of the figure presents the impact of our filtering technique on the Quick algorithm, on three metrics: runtime, size of the summary and its visibility computed as in Definition 3. The results prove the utility of our filtering technique with a significant decrease of the execution time, up to a factor 10 with 1750 nodes, while not impacting the compression rates nor the visibility. The right side of Fig. 2 presents a comparison of the three algorithms, with our filtering technique, on the same metrics. The results show that even if Algorithm 2 seems efficient regarding execution time and compression rates, the visibility obtained with this algorithm is worst than the others dropping around 0.6 while the others stay greater than 0.9. Finally, the greedy algorithm seems to perform surprisingly well on these graphs, being highly efficient in time and compression rate while keeping a good visibility.

5 Conclusion

In this paper, we focused on graph summarization using dense subgraphs and more precisely quasi-cliques. This approach requiring a quasi-clique mining algorithm, we show how to improve the already existing quasi-clique enumeration algorithms introducing a very effective pre-processing technique. Also, we adapt and introduce a quasi-clique mining algorithm designed to avoid redundancy while improving performances both in runtime and compression rate.

References

1. Ahmad, M., Beg, M.A., Khan, I., Zaman, A., Khan, M.A.: SsAG: Summarization and sparsification of attributed graphs (2021)
2. Baril, A., Dondi, R., Hosseinzadeh, M.M.: Hardness and tractability of the γ-complete subgraph problem. Inf. Process. Lett. **169**, 106105 (2021)
3. Boldi, P., Vigna, S.: The webgraph framework i: Compression techniques. In: Proceedings of WWW2004. pp. 595–602 (2004)
4. Lagraa, S., Seba, H., Khennoufa, R., MBaya, A., Kheddouci, H.: a distance measure for large graphs based on prime graphs. Pattern Recogn. **47**(9), 2993–3005 (2014)
5. Lee, K., Jo, H., Ko, J., Lim, S., Shin, K.: SSumM Sparse summarization of massive graphs. In: Proceedings of the 26th ACM SIGKDD. pp. 144–154 (2020)
6. Leskovec, J., Krevl, A.: SNAP datasets: stanford large network dataset collection (2014). http://snap.stanford.edu/data
7. Liu, G., Wong, L.: Effective pruning techniques for mining quasi-cliques. In: Daelemans, W., Goethals, B., Morik, K. (eds.) ECML PKDD 2008. LNCS (LNAI), vol. 5212, pp. 33–49. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-87481-2_3
8. Navlakha, S., Rastogi, R., Shrivastava, N.: Graph summarization with bounded error. In: Proceedings of the 2008 ACM SIGMOD Conference. pp. 419–432 (2008)
9. Wang, J., Cheng, J., Fu, A.W.C.: Redundancy-aware maximal cliques. In: Proceedings of the 19th ACM SIGKDD Conference. pp. 122–130 (2013)
10. Wang, L., Lu, Y., Jiang, B., Gao, K.T., Zhou, T.H.: Dense subgraphs summarization: an efficient way to summarize large scale graphs by super nodes. In: 16th International Conference on Intelligent Computing Methodologies, Italy. pp. 520–530 (2020)