K-Clique-Graphs for Dense Subgraph Discovery

Giannis Nikolentzos[1,2(✉)], Polykarpos Meladianos[1,2], Yannis Stavrakas[3], and Michalis Vazirgiannis[1,2]

[1] LIX, École Polytechnique, Palaiseau, France
[2] Athens University of Economics and Business, Athens, Greece
{nikolentzos,pmeladianos,mvazirg}@aueb.gr
[3] Institute for the Management of Information Systems RC "Athena", Athens, Greece
yannis@imis.athena-innovation.gr

Abstract. Finding dense subgraphs in a graph is a fundamental graph mining task, with applications in several fields. Algorithms for identifying dense subgraphs are used in biology, in finance, in spam detection, etc. Standard formulations of this problem such as the problem of finding the maximum clique of a graph are hard to solve. However, some tractable formulations of the problem have also been proposed, focusing mainly on optimizing some density function, such as the degree density and the triangle density. However, maximization of degree density usually leads to large subgraphs with small density, while maximization of triangle density does not necessarily lead to subgraphs that are close to being cliques.

In this paper, we introduce the k-clique-graph densest subgraph problem, $k \geq 3$, a novel formulation for the discovery of dense subgraphs. Given an input graph, its k-clique-graph is a new graph created from the input graph where each vertex of the new graph corresponds to a k-clique of the input graph and two vertices are connected with an edge if they share a common $k-1$-clique. We define a simple density function, the k-clique-graph density, which gives compact and at the same time dense subgraphs, and we project its resulting subgraphs back to the input graph. In this paper, we focus on the triangle-graph densest subgraph problem obtained for $k = 3$. To optimize the proposed function, we provide an exact algorithm. Furthermore, we present an efficient greedy approximation algorithm that scales well to larger graphs.

We evaluate the proposed algorithms on real datasets and compare them with other algorithms in terms of the size and the density of the extracted subgraphs. The results verify the ability of the proposed algorithms in finding high-quality subgraphs in terms of size and density. Finally, we apply the proposed method to the important problem of keyword extraction from textual documents. Code related to this chapter is available at: https://github.com/giannisnik/k-clique-graphs-dense-subgraphs.

Electronic supplementary material The online version of this chapter (https://doi.org/10.1007/978-3-319-71249-9_37) contains supplementary material, which is available to authorized users.

M. Ceci et al. (Eds.): ECML PKDD 2017, Part I, LNAI 10534, pp. 617–633, 2017.
https://doi.org/10.1007/978-3-319-71249-9_37

1 Introduction

In recent years, graph-based representations have become extremely popular for modelling real-world data. Some examples of data represented as graphs include social networks, protein or gene regulation networks and textual documents. The problem of extracting dense subgraphs from such graphs has received a lot of attention due to its potential applications in many fields. Specifically, in the web graph, dense subgraphs may correspond to link spam [18] and hence, they can be used for spam detection. In bioinformatics, they are used for finding molecular complexes in protein-protein interaction networks [6] and for discovering motifs in genomic DNA [17]. In the field of finance, they are used for discovering migration motifs in financial markets [14]. Other applications include graph compression [10], graph visualization [1], real-time identification of important stories in Twitter [3] and community detection [12].

Given an undirected, unweighted graph $G = (V, E)$, we will denote $|V| = n$ the number of vertices and $|E| = m$ the number of edges. Given a subset of vertices $S \subseteq V$, let $E(S)$ be the set of edges that have both end-points in S. Hence, $G(S) = (S, E(S))$ is the subgraph induced by S. The *density* of the set S is $\delta(S) = |E(S)|/\binom{|S|}{2}$, the number of edges in S over the total possible edges. Finding the set S that maximizes δ is not a meaningful problem, as density δ does not take into account the size of the subgraph. For example, a subgraph consisting of two vertices connected with an edge has higher density δ than a subgraph consisting of 100 vertices and all but one edge between them. However, clearly, we would prefer the latter subgraph from the former even if it achieves a lower value of density δ. Typically, the problem of dense subgraph discovery asks for a set of vertices S which is large and which has high density. Several different functions have been proposed in the literature that aim to solve this problem. Some of these functions can be optimized in polynomial time, however, most of these formulations of extracting dense subgraphs are NP-hard and also hard to approximate.

Recently, there was a growing interest in the extraction of subgraphs whose vertices are highly connected to each other [7,34,35]. However, existing methods do not always find subgraphs with high density δ. Instead, they prefer subgraphs with many vertices even if their density δ is not very high. In many cases, we are interested in discovering sets of vertices where there is an edge between almost all their pairs. In this paper, we introduce a new formulation for extracting dense subgraphs. We define a new family of functions for measuring the density of a subgraph and we provide exact and approximate algorithms that allow the extraction of large subgraphs with high density δ by maximizing these functions. Our contributions are fourfold:

(i) **New formulation:** We introduce the *k-clique-graph densest subgraph* (k-clique-GDS) problem, a new formulation for finding large subgraphs with high density δ. Given a value for k, we create a graph whose vertices correspond to k-cliques of the original graph and we draw edges between two k-cliques if they share a common $(k-1)$-clique. We then extract a dense subgraph from the new graph and we project the result back to the original graph. We focus on the special case obtained for $k = 3$ which we call the *triangle-graph densest subgraph* (TGDS) problem. We define a new density function which is suited to the needs of our problem.

(ii) **Exact algorithm:** We present an algorithm that solves exactly the TGDS problem. The algorithm finds the optimal subgraph by solving a series of supermodular maximization problems.

(iii) **Approximation algorithm:** We propose an efficient greedy approximation algorithm for the TGDS problem which removes one vertex at each iteration. The algorithm achieves nearly-optimal results on real-world networks.

(iv) **Experimental evaluation:** We evaluate our exact and approximation algorithms on several real-world networks. We compare the obtained subgraphs with those outputted by state-of-the-art algorithms and we observe that the proposed algorithms extract subgraphs of high quality. We also present an application of our problem to the task of keyword extraction from textual documents.

2 Related Work

In this section, we review the related work published in the areas of *Clique Finding*, *Dense Subgraph Discovery* and *Triangle Listing*.

Clique Finding. A clique is a graph whose vertices are all connected to each other. Hence, all cliques have density $\delta = 1$. A *maximum* clique of a graph is a clique, such that there is no clique with more vertices. Finding the maximum clique of a graph is an NP-complete problem [22]. The maximum clique problem is also hard to approximate. More specifically, Håstad showed in [20] that for any $\epsilon > 0$, there is no polynomial algorithm that approximates the maximum clique within a factor better than $\mathcal{O}(n^{1-\epsilon})$, unless NP has expected polynomial time algorithms. Feige presented in [15] a polynomial-time algorithm that approximates the maximum clique within a ratio of $\mathcal{O}(n(\log\log n)^2/(\log n)^3)$. A *maximal* clique is a clique that is not included in a larger clique. The Bron–Kerbosch algorithm is a recursive backtracking procedure [9] that lists all maximal cliques in a graph in $\mathcal{O}(3^{n/3})$ time.

Dense Subgraph Discovery. The problem of finding a dense subgraph given an input graph has been widely studied in the literature [24]. As mentioned above, such a problem aims at finding a subset of vertices $S \subseteq V$ of an input graph G that maximizes some notion of density. Among all the functions for evaluating dense subgraphs, degree density has gained increased popularity. The degree density of a set of vertices S is defined as $d(S) = 2|E(S)|/|S|$. The problem of finding the set of vertices that maximizes the degree density is known as the *densest subgraph* (DS) problem. The set of vertices $S \subseteq V$ that maximizes the degree density can be identified in polynomial time by solving a series of minimum-cut problems [19]. Charikar showed in [11] that the DS problem can also be formulated as a linear programming (LP) problem. In the same paper, the author proved that the greedy algorithm proposed by Asahiro et al. [5] provides a $\frac{1}{2}$-approximation to the DS problem in linear time.

Some variations of the DS problem include the *densest k-subgraph* (DkS), the *densest at-least-k-subgraph* (DalkS) and the *densest at-most-k-subgraph* (DamkS) problems. These variations put restrictions on the size of the extracted subgraph. The DkS identifies the subgraph with exactly k vertices that maximizes the degree density and is known to be NP-complete [4]. Feige et al. provided in [16] an approximation algorithm with approximation ratio $\mathcal{O}(n^{\delta})$, where $\delta < 1/3$. The DalkS and DamkS problems were introduced by Andersen and Chellapilla [2]. The first problem asks for the subgraph of highest degree density among all subgraphs with at least k vertices and is known to be NP-hard [23], while the second problem asks for the subgraph of highest density among all subgraphs with at most k vertices and is known to be NP-complete [2].

Tsourakakis introduced in [34] the *k-clique densest subgraph* (k-clique-DS) problem which generalizes the DS problem. The k-clique-DS problem maximizes the average number of k-cliques induced by a set $S \subseteq V$ over all possible vertex subsets. For $k = 3$, we obtain the so-called *triangle densest subgraph* (TDS) problem which maximizes the triangle density defined as $d_{tr}(S) = t(S)/|S|$ where $t(S)$ is the number of triangles in S. The author provides two polynomial-time algorithms that identify the exact set of vertices that maximizes the triangle density and a $\frac{1}{3}$-approximation algorithm which runs asymptotically faster than any of the exact algorithms.

There are several other recent algorithms that extract dense subgraphs by maximizing other notions of density [32,35,36]. It is worthwhile mentioning Tsourakakis et al.'s work [35]. The authors defined the *optimal quasi-clique* (OQC) problem which finds the subset of vertices $S \subseteq V$ that maximizes the function $f_{\alpha}(S) = |E(S)| - \alpha\binom{|S|}{2}$ where $\alpha \in (0, 1)$ is a constant. The OQC problem is not polynomial-time solvable and the authors provided a greedy approximation algorithm that runs in linear time and a local-search heuristic.

Triangle Listing. Given a graph G, the *triangle listing* problem reports all the triangles in G. The triangle listing problem has been extensively studied and a large number of algorithms has been proposed [13,21,30]. A listing algorithm requires at least one operation per triangle. In the worst case, there are n^3 triangles in terms of the number of vertices and $m^{3/2}$ in terms of the number of edges. Hence, in the worst case, it takes $m^{3/2}$ time just to report the triangles. The above algorithms require $\mathcal{O}(m^{3/2})$ time to list the triangles and they are thus optimal in the worst case. Recently, Björklund et al. proposed output sensitive algorithms which run asymptotically faster when the number of triangles in the graph is small [8].

3 Problem Definition

In this section, we will introduce the *k-clique-graph densest subgraph* (k-clique-GDS) problem, a novel formulation for finding dense subgraphs. In the following, we will restrict ourselves to the case where $k = 3$, that is to triangles. At the end

Algorithm 1. Construct triangle-graph

```
Input: graph $G = (V, E)$
Output: graph $G' = (V', E')$
  1: Assign a unique label to each edge of the input graph $G$.
  2: Extract all triangles in $G$ by running a triangle listing algorithm. Let $T(S)$ be the set of the extracted triangles.
  3: Create a new empty graph $G'$.
  4: For each triangle $t \in T(G)$ create a vertex in the $G'$.
  5: Connect two vertices in $G'$ with an edge if the corresponding triangles in $G$ share a common edge.
  6: Assign to the new edge the label of the edge that is shared between the two triangles.
  7: Return $G'$.
```

of the section, we will describe how the proposed approach can be generalized to the case of k-cliques, $k > 3$.

The cornerstone of the proposed method is the transformation of the input graph $G = (V, E)$ into another graph $G' = (V', E')$. The transformed graph G' is a more abstract representation of G. Specifically, it encodes information regarding the triangles of the input graph G and the relationships between them.

As a preprocessing step before applying the transformation, we assign labels to the edges of the input graph G. Given a set of labels $L, \ell : E \rightarrow L$ is a function that assigns labels to the edges of the graph. Each edge is assigned a unique label. Hence, the cardinality of the set L is equal to that of set $E, |L| = |E|$. We next proceed with the transformation of G into G'. The first step of the transformation procedure is to run a triangle listing algorithm. There are several available triangle listing algorithms as described in Sect. 2. Let $T(S)$ be the set of triangles extracted from G. For each triangle $t \in T(G)$, we create a vertex in the new graph G'. Therefore, each vertex represents one of the triangles extracted from G. Pairs of triangles that share a common edge in G are considered neighbors and are connected with an edge in G'. In other words, each edge in G' corresponds to a pair of triangles sharing the same edge. The edges of G' are also assigned labels. Each edge in G' is given the label of the edge that is shared between the two corresponding triangles in G. For example, given a pair of triangles $t_1 = (v_1, v_2, v_3)$ and $t_2 = (v_1, v_2, v_4)$ where $t_1, t_2 \in T(G)$, these triangles have a common edge $e = (v_1, v_2)$ and the edge e' that links them in G' gets the same label as e, that is $\ell(e') = \ell(e)$. A triangle has three edges, hence, although it can have any number of adjacent edges in G', its labels come from a limited alphabet consisting of only three items (the labels of the three edges of the triangle in G). We call the transformed graph G' the *triangle-graph* of G. Algorithm 1 describes the steps required to create G' from G and Fig. 1 illustrates how a graph containing 4 triangles is transformed into its triangle-graph.

After creating the triangle-graph G', we can find a subset of vertices $S' \subset V'$ that correponds to a dense subgraph. As mentioned earlier, each vertex $v \in S'$ represents a triangle t of the input graph G. Each triangle t is a set of three vertices. Intuitively, the union of the vertices of all the triangles that belong to the set S' will form a dense subgraph of G. To extract the set of vertices S', we can define a density measure and optimize it. A simple measure we can employ is the well-known degree density defined as $d(S') = 2|E(S')|/|S'|$. However, the above function will not necessarily lead to subgraphs with high density. Consider the two graphs shown in Fig. 2. As can be seen from the Figure, the triangle-graphs emerging from the two input graphs are structurally equivalent, and hence, they have the same degree density. As a result, if the two graphs are components of a larger graph and there are no other subgraphs with higher value, they are equally likely solutions to the DS problem. However, it is obvious that the upper graph suits better our purpose, and we would like our algorithm to prefer this compared to the lower graph.

To account for this problem, we define a new density measure which we call the *triangle-graph density*.

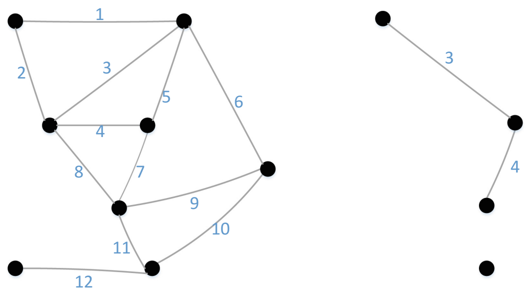

Fig. 1. Example of an input graph (left) and the triangle-graph (right) created from it. There are 4 triangles in the input graph defined by the following triads of edges: $(1, 2, 3)$, $(3, 4, 5)$, $(4, 7, 8)$ and $(9, 10, 11)$. The first two as well as the second and third triangles have a common edge (edge 3 and edge 4 respectively). Hence, these pairs of triangles are connected with an edge in the triangle-graph. The fourth triangle does not share any edges with the other triangles, therefore, it has no adjacent edges in the triangle-graph.

Definition 1 (Triangle-Graph Density). *Given an undirected, unweighted graph $G = (V, E)$, first construct its triangle graph $G' = (V', E')$. For any $S' \subseteq V'$, we define its triangle-graph density as $f(S') = \frac{d(S')}{|S'|}$ where $d(S') = \sum_{v \in S'} \min_{l \in L(v)} \big(deg_{S'}(v, l)\big), L(v)$ the set of labels of the edges adjacent to v (three labels at most), and $deg_{S'}(v, l)$ the number of edges that are adjacent to v in the subgraph induced by S' and are assigned the label l.*

The triangle-graph density will allow the discovery of subgraphs with high values of density δ. This is due to the fact that for each triangle t in G, the function takes into account the number of neighbors from all three edges of t. If a triangle t corresponding to the vertex v in G' shares one of its edges with many other triangles,

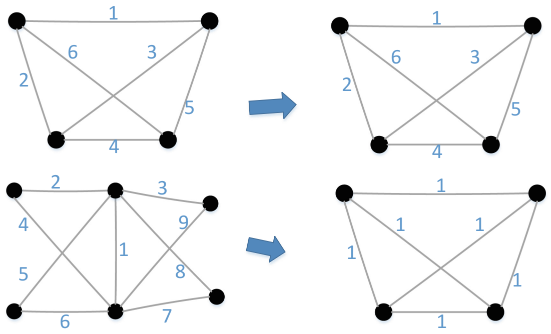

Fig. 2. Two input graphs (left) and their triangle-graphs (right). The two triangle-graphs are structurally equivalent although the input graphs are not.

but the other two edges with no triangles, then $\min_{l\in L(v)}\big(deg_{S'}(v,l)\big) = 0$. Therefore, even if t has many neighbors, it contributes nothing to the triangle-graph density. Triangle-graph density seeks for subgraphs whose vertices belong to edges which all consist of large sets of vertices. Cliques are natural candidates for maximizing the function since all their edges are shared between several triangles.

We next introduce the *triangle-graph densest subgraph* problem, the optimization problem we address in this paper.

Problem 1 (TGDS problem). Given an undirected, unweighted graph $G = (V, E)$, create its triangle-graph $G' = (V', E')$, and find a subset of vertices $S^* \subseteq V'$ such that $f(S^*) = \arg\max_{S'\subseteq V'} f(S')$.

After optimizing the triangle-graph density, we end up with a set of vertices $S' \subseteq V'$ and from these we obtain the set of vertices $S \subseteq V$ that corresponds to the resulting subgraph. The set S consists of all the vertices that form the triangles in S'. It is clear that the TGDS problem can result in subgraphs with high values of density δ.

What needs to be investigated next is what are the properties of the extracted subgraphs and how they differ from the ones extracted from existing methods. The proposed *triangle-graph densest subgraph* (TGDS) problem seems to be very related to the *triangle densest subgraph* (TDS) problem introduced by Tsourakakis in [34]. However, as we will show next, the two problems can result in different solutions, and the subgraphs returned by TGDS are closer to being near-cliques compared to the ones returned by TDS. Consider the graph G and its triangle-graph G' both shown in Fig. 3. The optimal solution of TDS is the whole graph. Conversely, the optimal solution of TGDS is the subgraph induced by the vertices that form the 4-clique. Hence, the optimal solution of the proposed problem is a clique, while the optimal solution of TDS is a larger graph with lower density δ. The above example demonstrates that the optimal solutions of TGDS correspond to subgraphs that exhibit a stronger near-clique structure compared to TDS.

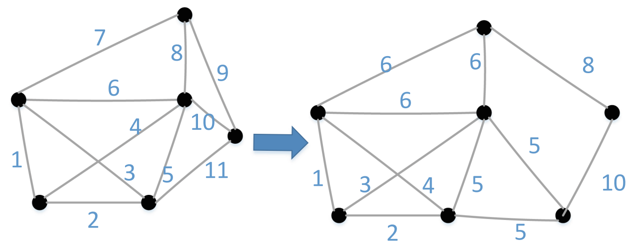

Fig. 3. Example of an input graph (left) and the triangle-graph (right) created from it. There are 7 triangles in the input graph defined by the following triads of edges: $(1, 2, 3)$, $(1, 4, 6)$, $(2, 4, 5)$, $(3, 5, 6)$, $(6, 7, 8)$, $(8, 9, 10)$ and $(5, 10, 11)$.

The process of creating the k-clique graph for $k > 3$ is similar to the one described above for $k = 3$. Specifically, to construct the k-clique graph $G' = (V', E')$, we first extract all the k-cliques from G. Then for each k-clique in G, we create a vertex v in G'. Two vertices $v_1, v_2 \in V'$ are connected with an edge if the corresponding cliques share a common $(k-1)$-clique in G. For example, for $k = 4$, if two 4-cliques in G share a common triangle, an edge is drawn between them in G'. Each $(k-1)$-clique in G is assigned a unique label and the edges of the k-clique graph are assigned the labels of the $(k-1)$-cliques that are shared between their two endpoints. Then, the *k-clique-graph density* and the *k-clique-graph densest subgraph* (k-clique-GDS) problem are defined in a similar way as in the case of triangles. The algorithms presented in the next section for maximizing triangle-graph density can be generalized to maximizing the k-clique-graph density. However, extracting k-cliques for $k > 3$ is a computationally demanding task, and hence, we restrict ourselves to the case where $k = 3$.

4 Proposed Methods

In this section, we present some algorithms for solving the TGDS problem. These algorithms are inspired by previously-introduced algorithms in the field of dense subgraph discovery. More specifically, we provide an algorithm that solves the TGDS problem exactly as well as a greedy approximation algorithm. In what follows, we assume that we have extracted all triangles from the input graph and we have created the triangle-graph. Note that, for simplicity of notation, from now on, we denote by $G = (V, E)$ the triangle-graph and not the input graph. We also denote by $q_S(v)$ the minimum degree of vertex v with respect to the three labels of its adjacent edges in the subgraph induced by S, that is $q_S(v) = \min_{l \in L(v)} \big(deg_S(v, l)\big)$.

4.1 A Supermodular Maximization Approach

In this section, we provide an exact algorithm for finding the set of vertices S^* that maximizes the triangle-graph density. The algorithm is based on the *supermodular maximization* approach proposed by Tsourakakis in [34]. More specifically, maximizing the triangle-graph density can be cast as a supermodular maximization problem. We next introduce a brief background on submodularity and supermodularity.

Submodular and supermodular functions are classes of functions with many useful properties which have found application in several real world problems. The main property of supermodular functions is that given two sets A and B, where $A \subseteq B \subseteq V \backslash v$, the difference in the incremental value of the function that a single element v makes when added to an input set increases as the size of the input set increases. Hence, the incremental value of adding v to sets A and B is larger for B compared to A. Let V be a finite ground set. A function $h : 2^V \rightarrow \mathcal{R}$ that maps subsets $S \subseteq V$ to a real value $h(S)$ is called *supermodular* if the following equation holds for any $A, B \subseteq V$

$$h(A \cup B) + h(A \cap B) \geq h(A) + h(B)$$

We next give a second equivalent definition of supermodularity. Let A, B be two sets such that $A \subseteq B \subseteq V$ and $v \in V \backslash B$. Then, function h is supermodular if

$$h(B \cup \{v\}) - h(B) \geq h(A \cup \{v\}) - h(A)$$

This second form of supermodularity shows that the value added by a new element v never decreases when the context gets larger. Submodular and supermodular functions have gained increased popularity due to the fact that they can be minimized and maximized respectively in strongly polynomial time [31]. More specifically, Orlin proposed in [28] an algorithm for maximizing an integer valued supermodular function f which runs in $\mathcal{O}(n^5 EO + n^6)$ where EO is the time to evaluate $h(S)$ for some $S \subseteq V$.

We next show that for any $\alpha \in \mathcal{R}^+$, the function $h : 2^V \rightarrow \mathcal{R}$ defined by $h(S) = d(S) - \alpha|S|$ is supermodular.

Theorem 1. *Function $h : 2^V \rightarrow \mathcal{R}$ where $h(S) = d(S) - \alpha|S|$ is supermodular.*

Proof. The proof is left to the supplementary material [27].

To find the set of vertices S^* that maximizes the triangle-graph density, we can use Algorithm 2. The algorithm terminates in logarithmic number of rounds. In each iteration, we run the Orlin-Supermodular-Opt procedure in order to find the set of vertices that maximize function h given the current value of parameter α.

Theorem 2. *There exists an algorithm that solves the TGDS problem and runs in $\mathcal{O}\Big(m^{3/2} + \big(t^5(t+y) + t^6\big)\log t\Big)$ time where m is the number of edges of the input graph, t is the number of triangles in the input graph and y is the number of edges of the triangle-graph.*

Algorithm 2. Supermodular maximization algorithm

```
Input: triangle-graph $G = (V, E)$
Output: Subset of vertices $S^* \subseteq V$
  $l \leftarrow \frac{d(V)}{n}$
  $u \leftarrow \frac{n(n-1)}{3n}$
  $S^* \leftarrow V$
  while $u - l \geq \frac{1}{n(n-1)}$ do
      $\alpha \leftarrow \frac{l+u}{2}$
      $(val, S) \leftarrow$ Orlin-Supermodular-Opt$(G, \alpha)$
      if $val < 0$ then
          $u \leftarrow \alpha$
      else
          $l \leftarrow \alpha$
          $S^* \leftarrow S$
      end if
  end while
  Return $S^*$
```

To create the triangle-graph from the input graph, we first need to run a triangle listing algorithm. The one proposed by Itai and Rodeh runs in $\mathcal{O}(m^{3/2})$ time [21]. As mentioned above, the algorithm will run in a logarithmic number of rounds. Furthermore, Orlin's algorithm runs in $\mathcal{O}(n^5 EO + n^6)$ time where n is the size of the ground set and EO is the time to evaluate $h(S)$ for some $S \subseteq V$ [28]. In our case, the ground set corresponds to the vertices of the triangle-graph. Hence, it is equal to the number of triangles t in the input graph. As regards the complexity of computing $h(S)$, it is linear to the number of vertices and number of edges of the triangle-graph. Let y denote the number of edges of the triangle-graph. The overall running time of Algorithm 2 is thus $\mathcal{O}\Big(m^{3/2} + \big(t^5(t + y) + t^6\big) \log t\Big)$.

Lemma 1. *Algorithm 2 solves the TGDS problem and runs in* $\mathcal{O}\Big(m^{3/2} + \big(t^5(t + y) + t^6\big) \log t\Big)$ *time.*

Proof. The proof is left to the supplementary material [27].

4.2 A Greedy Approximation Algorithm

In this section, we provide an efficient algorithm for extracting a set of vertices $S \subseteq V$ with high value of triangle-graph density $f(S)$. The proposed algorithm is an adaptation of the greedy algorithm of Asahiro et al. [5]. The algorithm is illustrated as Algorithm 3. The algorithm iteratively removes the vertex v whose value $d(v)$ is the smallest among all vertices. Subsequently, it computes the triangle-graph density of the subgraph induced by the remaining vertices. The output is the subgraph over all the produced subgraphs that maximizes triangle-graph density. The algorithm is linear to the number of vertices and the

Algorithm 3. Greedy algorithm

```
Input: graph $G = (V, E)$
Output: Subset of vertices $S \subseteq V$
  $S_{|V|} \leftarrow V$
  for $i \leftarrow |V|$ to 1 do
      Let $v$ be the vertex whose minimum value of the three degrees is the smallest in the subgraph induced by $S_i$
      $S_{i-1} \leftarrow S_i \setminus \{v\}$
  end for
  $S \leftarrow \arg\max_{i=1,\ldots,|V|} f(S_i)$
```

number of edges of the triangle-graph, hence its complexity is $\mathcal{O}(t + y)$ where t is the number of triangles in the input graph and y is the number of edges of the triangle-graph.

Theorem 3. *Let S be the set of vertices returned after the execution of Algorithm 3 and let S^* be the set of vertices of the optimal subgraph. Consider the iteration of the greedy algorithm just before the first vertex u that belongs in the optimal set S^* is removed, and let S_I denote the vertex set currently kept in that iteration. Let also $q_{S_I}(u)$ be the minimum degree of vertex u in S_I with respect to the three labels of its adjacent edges. Then, it holds that*

$$f(S) \geq \frac{|S^*|}{|S_I|} f_G^* + \left(1 - \frac{|S^*|}{|S_I|}\right) q_{S_I}(u)$$

Proof. The proof is left to the supplementary material [27].

From the above result, we can see that the bound provided by the approximation algorithm highly depends on the relationship between $|S_I|$, the size of the vertex set just before the first vertex of S^* is removed, and $|S^*|$, the size of the optimal set. It also depends on the relationship between the optimal value of the triangle-graph density $f(S^*)$ and the minimum degree $q_{S_I}(u)$ of the first vertex of the optimal set S^* to be removed from S_I with respect to its three labels. The difference between $|S_I|$ and $|S^*|$, and between $f(S^*)$ and $q_{S_I}(u)$ is not very large in practice, and the algorithm leads to subgraphs with quality almost equal to that of the optimal subgraphs.

5 Experiments and Evaluation

In this section, we present the evaluation of the proposed approach for extracting dense subgraphs. We first give details about the datasets that we used for our experiments. We then present the employed experimental settings. And we last report on the results obtained by our approach and some other methods.

5.1 Experimental Setup

For the evaluation of the proposed algorithms, we employed several publicly available graphs. The algorithms are applicable to simple unweighted, undirected graphs. Hence, we made all graphs simple by ignoring the edge direction in the case of directed graphs and by removing self-loops and edge weights, if any. Table 1 shows statistics of these graphs. The first ten datasets were obtained from UCIrvine Network Data Repository[1], while the remaining datasets were obtained from Stanford SNAP Repository[2]. We compared the proposed algorithms with algorithms that solve the *densest subgraph* (DS), the *triangle densest subgraph* (TDS) and the *optimal quasi-clique* (OQC) problems. For the first two (DS and TDS problems) as well as for the proposed problem, there are algorithms that solve these problems exactly in polynomial time. Hence, for small-sized datasets, we present the results obtained from both the exact and greedy approximation algorithms for each problem. For larger datasets, we report only on the results achieved by the greedy approximation algorithms. With regards to the objective function of the OQC problem, we set the value of parameter α equal to $1/3$ as suggested in [35]. All algorithms were implemented in Python[3] and all experiments were conducted on a single machine with a 3.4 GHz Intel Core i7 processor and 32 GB of RAM. To assess the quality of the extracted subgraphs, we employed the following measures: the density of the extracted subgraph $\delta(S) = |E(S)|/\binom{|S|}{2}$, the density with respect to the number of triangles $\tau(S) = t(S)/\binom{|S|}{3}$, that is the number of triangles in S over the total possible triangles, and the size of the subgraph $|S|$. The δ and τ measures take values between 0 and 1. The larger their value, the closer the subgraph to being a clique. Therefore, we are interested in finding large subgraphs (large value of $|S|$) with δ and τ values close to 1.

5.2 Results and Discussion

Table 2 summarizes the results obtained on small-sized graphs. We observe that on the small-sized graphs, the proposed algorithms (Exact TGDS and Greedy TGDS) return in general subgraphs that are closer to being a clique compared to the competing algorithms. As we can see from the Table, the densities δ and τ of the subgraphs extracted by our algorithms are relatively high. Our initial intention was to design an algorithm for finding a set of vertices with many edges between them. The obtained results verify our intuition that the proposed approach is capable of finding near-cliques. Furthermore, we show in Table 3 the triangle-graph density of the subgraphs extracted by the exact and the greedy approximation algorithm. We notice that on four out of the six graphs, the two densities are equal to each other, while on the other two, they are very close

[1] https://networkdata.ics.uci.edu/index.php.
[2] http://snap.stanford.edu/data/index.html.
[3] Code is available at https://github.com/giannisnik/k-clique-graphs-dense-subgraphs.

Table 1. Graphs used for evaluating the algorithms.

Graph	$\|V\|$	$\|E\|$
Karate	34	78
Dolphins	62	159
Lesmis	77	254
Adjnoun	112	425
Football	115	613
Polbooks	105	441
Celegansneural	297	2,148
Polblogs	1,224	16,715
Power	4,941	6,594
Wiki-Vote	7,115	100,762
ca-CondMat	23,133	93,439
p2p-Gnutella31	62,586	147,892
Slashdot0902	82,168	504,230
email-EuAll	265,009	364,481
web-NotreDame	325,729	1,497,134
Amazon	334,863	925,872
Youtube	1,134,890	2,987,624
roadNet-CA	1,965,206	2,766,607

Table 2. Comparison of the extracted subgraphs by Goldberg's exact algorithm for the DS problem (Exact DS), Charikar's $\frac{1}{2}$ approximation algorithm for the DS problem (Greedy DS), Tsourakakis's algorithm for the TDS problem (Exact TDS), Tsourakakis's $\frac{1}{3}$ approximation algorithm for the TDS problem (Greedy TDS), Tsourakakis et al.'s greedy approximation algorithm for the OQC problem (Greedy OQC), our exact algorithm for the TGDS problem (Exact TGDS), and our greedy approximation algorithm for the TGDS problem (Greedy TGDS).

Dataset	Exact DS			Greedy DS			Exact TDS			Greedy TDS			Greedy OQC			Exact TGDS			Greedy TGDS		
	$\|S\|$	δ	τ	$\|S\|$	δ	τ	$\|S\|$	δ	τ	$\|S\|$	δ	τ	$\|S\|$	δ	τ	$\|S\|$	δ	τ	$\|S\|$	δ	τ
Karate	16	0.35	0.05	16	0.35	0.05	6	0.93	0.80	6	0.93	0.80	10	0.55	0.18	6	0.93	0.80	6	0.93	0.80
Dolphins	20	0.32	0.04	36	0.17	0.01	7	0.80	0.54	6	0.93	0.80	13	0.47	0.11	6	0.93	0.80	6	0.93	0.80
Lesmis	23	0.49	0.18	23	0.49	0.18	13	0.88	0.71	13	0.88	0.71	22	0.50	0.19	12	0.93	0.83	12	0.93	0.83
Adjnoun	48	0.20	0.01	44	0.22	0.01	41	0.23	0.01	41	0.23	0.01	16	0.48	0.11	8	0.82	0.51	7	0.85	0.62
Football	115	0.09	0.00	115	0.09	0.00	18	0.48	0.20	18	0.48	0.20	10	0.88	0.66	18	0.48	0.20	18	0.48	0.20
Polbooks	24	0.41	0.09	48	0.19	0.02	20	0.49	0.15	36	0.26	0.04	14	0.67	0.30	16	0.59	0.23	13	0.69	0.34

to each other. The obtained results indicate that the greedy algorithm achieves approximation ratios close to 1 on real-world networks. Hence, the approximation algorithm is nearly-optimal in practice.

Next, we present results obtained on larger graphs. Specifically, Table 4 compares the four approaches on 12 graphs. In general, the proposed algorithm still manages to extract subgraphs with high values of δ and τ. However, on two

Table 3. Triangle-graph densities of the subgraphs extracted by the exact and the greedy approximation algorithms.

Dataset	Exact TGDS	Greedy TGDS
Karate	2.25	2.25
Dolphins	2.25	2.25
Lesmis	7.60	7.60
Adjnoun	2.39	2.36
Football	6.0	6.0
Polbooks	4.02	3.89

Table 4. Comparison of the extracted subgraphs by Charikar's $\frac{1}{2}$ approximation algorithm for the DS problem (Greedy DS), Tsourakakis's $\frac{1}{3}$ approximation algorithm for the TDS problem (Greedy TDS), Tsourakakis et al.'s greedy approximation algorithm for the OQC problem (Greedy OQC), and our greedy approximation algorithm for the TGDS problem (Greedy TGDS).

Dataset	Greedy DS			Greedy TDS			Greedy OQC			Greedy TGDS		
	$\|S\|$	δ	τ	$\|S\|$	δ	τ	$\|S\|$	δ	τ	$\|S\|$	δ	τ
Celegansneural	127	0.13	0.005	30	0.47	0.13	22	0.61	0.25	24	0.55	0.21
Polblogs	278	0.20	0.020	102	0.54	0.195	100	0.55	0.202	74	0.67	0.343
Power	31	0.20	0.021	12	0.54	0.195	12	0.54	0.195	12	0.54	0.195
Wiki-Vote	828	0.11	0.004	464	0.19	0.014	133	0.47	0.131	152	0.42	0.104
ca-CondMat	26	1.0	1.0	26	1.0	1.0	26	1.0	1.0	26	1.0	1.0
p2p-Gnutella31	1,549	0.005	0.0	10	0.40	0.11	14	0.48	0.0	22	0.15	0.016
soc-Slashdot0902	219	0.39	0.097	171	0.50	0.165	155	0.54	0.200	145	0.56	0.225
email-EuAll	505	0.13	0.005	200	0.29	0.041	97	0.51	0.164	91	0.52	0.179
web-NotreDame	1,367	0.11	0.012	457	0.34	0.114	305	0.51	0.255	155	1.0	1.0
Amazon	9	0.91	0.761	16	0.45	0.178	9	0.91	0.761	170	0.03	0.001
Youtube	1,860	0.049	0.0006	729	0.11	0.005	125	0.46	0.115	442	0.17	0.012
roadNet-CA	19,899	0.0001	0.0	168	0.017	0.0002	5	0.80	0.40	168	0.017	0.0002

graphs (Amazon, roadNet-CA), it fails to discover high-quality subgraphs in terms of density. Overall, the Greedy DS algorithm returns the largest subgraphs, followed by the Greedy TDS algorithm, while the Greedy OQC algorithm and the proposed algorithm return smaller subgraphs with higher values of density. We notice that the subgraphs extracted by the proposed greedy approximation algorithm resemble most those extracted by the Greedy TDS algorithm. On the ca-CondMat dataset, all the algorithms extract the same subgraph.

6 Application

In this section, we apply the proposed algorithm to a central problem in Natural Language Processing: extracting keywords from a textual document. Keyword extraction finds applications in several fields from information retrieval to text classification and summarization. Given a document d, we can represent it as a statistical *graph-of-words*, following earlier approaches in keyword extraction [26,29,33] and in summarization [25]. The construction of the graph is preceded by a preprocessing phase where standard text processing tasks are performed. The processed document is then transformed into an unweighted, undirected graph G whose vertices represent unique terms and whose edges represent co-occurrences between the connected terms within a fixed-size window. We then employ the proposed algorithm to extract a dense subgraph from G. The vertices of the subgraph act as representative keywords of the document.

To demonstrate the ability of the proposed approach to identify meaningful keywords, we extracted the text of this paper and we transformed it into a graph G using a window of size 3 (each word is connected with an edge with each one of its two preceding and two following words, if any). We then extracted a dense subgraph from G using the proposed greedy approximation algorithm. The output subgraph consists of the following 25 vertices:

```
set, labels, number, subgraphs, triangles, maximizes, given,
density, graph, input, function, triangle, subgraph, cliques,
vertex, edges, clique, algorithm, k, vertices, value, edge,
supermodular, hence, problem
```

As we can observe, the extracted keywords capture the main concepts of the paper.

7 Conclusion

In this paper, we propose a novel approach for extracting dense subgraphs. Given a graph, our algorithm first transforms it to a k-clique-graph. We then introduce a simple density measure to extract high-quality subgraphs. We propose an algorithm for exactly maximizing the density function. We also present a greedy approximation algorithm. We evaluate our proposed approach for the case where $k = 3$ on real graphs and we compare it with other popular measures. Overall, our algorithms show good performance in finding large near-cliques, and can serve as useful additions to the list of dense subgraph discovery algorithms.

References

1. Alvarez-Hamelin, J.I., Dall'Asta, L., Barrat, A., Vespignani, A.: Large scale networks fingerprinting and visualization using the k-core decomposition. In: NIPS 2005, pp. 41–50 (2005)
2. Andersen, R., Chellapilla, K.: Finding dense subgraphs with size bounds. In: Avrachenkov, K., Donato, D., Litvak, N. (eds.) WAW 2009. LNCS, vol. 5427, pp. 25–37. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-540-95995-3_3
3. Angel, A., Koudas, N., Sarkas, N., Srivastava, D., Svendsen, M., Tirthapura, S.: Dense subgraph maintenance under streaming edge weight updates for real-time story identification. VLDB J. **23**(2), 175–199 (2014)

4. Asahiro, Y., Hassin, R., Iwama, K.: Complexity of finding dense subgraphs. Discret. Appl. Math. **121**(1), 15–26 (2002)
5. Asahiro, Y., Iwama, K., Tamaki, H., Tokuyama, T.: Greedily finding a dense subgraph. J. Algorithms **34**(2), 203–221 (2000)
6. Bader, G.D., Hogue, C.W.: An automated method for finding molecular complexes in large protein interaction networks. BMC Bioinform. **4**(1), 1 (2003)
7. Balalau, O.D., Bonchi, F., Chan, T., Gullo, F., Sozio, M.: Finding subgraphs with maximum total density and limited overlap. In: WSDM 2015, pp. 379–388 (2015)
8. Björklund, A., Pagh, R., Williams, V.V., Zwick, U.: Listing triangles. In: Esparza, J., Fraigniaud, P., Husfeldt, T., Koutsoupias, E. (eds.) ICALP 2014. LNCS, vol. 8572, pp. 223–234. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-43948-7_19
9. Bron, C., Kerbosch, J.: Algorithm 457: finding all cliques of an undirected graph. Commun. ACM **16**(9), 575–577 (1973)
10. Buehrer, G., Chellapilla, K.: A scalable pattern mining approach to web graph compression with communities. In: WSDM 2008, pp. 95–106 (2008)
11. Charikar, M.: Greedy approximation algorithms for finding dense components in a graph. In: Jansen, K., Khuller, S. (eds.) APPROX 2000. LNCS, vol. 1913, pp. 84–95. Springer, Heidelberg (2000). https://doi.org/10.1007/3-540-44436-X_10
12. Chen, J., Saad, Y.: Dense subgraph extraction with application to community detection. TKDE **24**(7), 1216–1230 (2012)
13. Chiba, N., Nishizeki, T.: Arboricity and subgraph listing algorithms. In: SICOMP 1985, vol. 14, no. 1, pp. 210–223 (1985)
14. Du, X., Jin, R., Ding, L., Lee, V.E., Thornton Jr., J.H.: Migration motif: a spatial-temporal pattern mining approach for financial markets. In: KDD 2009, pp. 1135–1144 (2009)
15. Feige, U.: Approximating maximum clique by removing subgraphs. In: SIDMA 2004, vol. 18, no. 2, pp. 219–225 (2004)
16. Feige, U., Peleg, D., Kortsarz, G.: The dense k-subgraph problem. Algorithmica **29**(3), 410–421 (2001)
17. Fratkin, E., Naughton, B.T., Brutlag, D.L., Batzoglou, S.: MotifCut: regulatory motifs finding with maximum density subgraphs. Bioinformatics **22**(14), e150–e157 (2006)
18. Gibson, D., Kumar, R., Tomkins, A.: Discovering large dense subgraphs in massive graphs. In: VLDB 2005, pp. 721–732 (2005)
19. Goldberg, A.V.: Finding a maximum density subgraph. Technical report, University of California Berkeley (1984)
20. Håstad, J.: Clique is hard to approximate within $n^{1-\epsilon}$. In: FOCS 1996, pp. 627–636 (1996)
21. Itai, A., Rodeh, M.: Finding a minimum circuit in a graph. In: SICOMP 1978, vol. 7, no. 4, pp. 413–423 (1978)
22. Karp, R.M.: Reducibility Among Combinatorial Problems. Springer, Boston (1972). https://doi.org/10.1007/978-1-4684-2001-2_9
23. Khuller, S., Saha, B.: On finding dense subgraphs. In: Albers, S., Marchetti-Spaccamela, A., Matias, Y., Nikoletseas, S., Thomas, W. (eds.) ICALP 2009. LNCS, vol. 5555, pp. 597–608. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-02927-1_50
24. Lee, V.E., Ruan, N., Jin, R., Aggarwal, C.: A survey of algorithms for dense subgraph discovery. In: Managing and Mining Graph Data, pp. 303–336 (2010)

25. Meladianos, P., Nikolentzos, G., Rousseau, F., Stavrakas, Y., Vazirgiannis, M.: Degeneracy-based real-time sub-event detection in Twitter stream. In: ICWSM 2015, pp. 248–257 (2015)
26. Mihalcea, R., Tarau, P.: TextRank: bringing order into texts. In: EMNLP 2004, pp. 404–411 (2004)
27. Nikolentzos, G., Meladianos, P., Stavrakas, Y., Vazirgiannis, M.: Supplementary material for k-clique-graphs for dense subgraph discovery (2017). http://www.db-net.aueb.gr/nikolentzos/files/ecml_pkdd17_suppl.pdf
28. Orlin, J.B.: A faster strongly polynomial time algorithm for submodular function minimization. Math. Program. **118**(2), 237–251 (2009)
29. Rousseau, F., Vazirgiannis, M.: Main core retention on graph-of-words for single-document keyword extraction. In: Hanbury, A., Kazai, G., Rauber, A., Fuhr, N. (eds.) ECIR 2015. LNCS, vol. 9022, pp. 382–393. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-16354-3_42
30. Schank, T., Wagner, D.: Finding, counting and listing all triangles in large graphs, an experimental study. In: Nikoletseas, S.E. (ed.) WEA 2005. LNCS, vol. 3503, pp. 606–609. Springer, Heidelberg (2005). https://doi.org/10.1007/11427186_54
31. Schrijver, A.: A combinatorial algorithm minimizing submodular functions in strongly polynomial time. JCT **80**(2), 346–355 (2000)
32. Sozio, M., Gionis, A.: The community-search problem and how to plan a successful cocktail party. In: KDD 2010, pp. 939–948 (2010)
33. Tixier, A.J.P., Malliaros, F.D., Vazirgiannis, M.: A graph degeneracy-based approach to keyword extraction. In: EMNLP 2016 (2016)
34. Tsourakakis, C.: The k-clique densest subgraph problem. In: WWW 2015, pp. 1122–1132 (2015)
35. Tsourakakis, C., Bonchi, F., Gionis, A., Gullo, F., Tsiarli, M.: Denser than the densest subgraph: extracting optimal quasi-cliques with quality guarantees. In: KDD 2013, pp. 104–112 (2013)
36. Wang, N., Zhang, J., Tan, K.L., Tung, A.K.: On triangulation-based dense neighborhood graph discovery. VLDB Endow. **4**(2), 58–68 (2010)