K-clique-graphs for Dense Subgraph Discovery

Giannis Nikolentzos  
AUEB and École Polytechnique  
nikolentzos@aueb.gr

Polykarpos Meladianos  
AUEB and École Polytechnique  
pmeladianos@aueb.gr

Michalis Vazirgiannis  
École Polytechnique and AUEB  
mvazig@lix.polytechnique.fr

Yannis Stavarakas  
IMIS / RC ATHENA  
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. However, maximization of degree density usually leads to large subgraphs with small density.

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 two exact algorithms. 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.

Keywords
Graph mining; Dense subgraph discovery; Near-clique extraction

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 $\delta$ is not a meaningful problem, as density $\delta$ 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 $\delta$ 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 $\delta$. 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 [31 7 30]. However, existing methods do not always find...
subgraphs with high density \(\delta\). Instead, they prefer subgraphs with many vertices even if their density \(\delta\) 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 \(\delta\) 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 \(\delta\). 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 algorithms:** We present two algorithms that solve exactly the TGDS problem. The first finds the optimal subgraph by solving a series of supermodular maximization problems, while the second is based on linear programming. Specifically, it formulates the problem as a linear program and computes the optimal subgraph based on the optimal solution to this linear program.

(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.

**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 \[23\]. The maximum clique problem is also hard to approximate. More specifically, Håstad showed in \[21\] that for any \(\epsilon > 0\), there is no polynomial algorithm that approximates the maximum clique within a factor better than \(O(n^{1-\epsilon})\), unless \(NP\) has expected polynomial time algorithms. Feige presented in \[15\] a polynomial-time algorithm that approximates the maximum clique within a ratio of \(O(n^{(\log \log n)^2/(\log n)^3})\). 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 \(O(3^n/3)\) time.

**Dense Subgraph Discovery.** The problem of finding a dense subgraph given an input graph has been widely studied in the literature \[25\]. 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 \(d(S) = |E(S)|/|S|\). 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 \(1/2\)-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 \(O(n^{1/2})\), 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 \[24\], 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 \[30\] 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 \(d_3(S) = |E(S)|/|S|^3\) where \(|S|\) 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 \(1/2\)-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 \[29\ 31\ 32\]. It is worthwhile mentioning Tsourakakis et al.’s work \[31\]. The authors defined the optimal quasi-clique (OQC) problem which finds the subset of vertices \(S \subseteq V\) that maximizes the function \(f_\alpha(S) = |E(S)| - \alpha |S|^2\) 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 \[22\ 13\ 27\]. 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 tri-
angles. The above algorithms require $O(m^{3/2})$ 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 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 which can be employed as described in Section 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. Algorithm 1 describes the steps required to create $G'$ from $G$ and Figure 1 illustrates how a graph containing 4 triangles is transformed into its triangle hypergraph.

After creating the triangle-graph $G'$, we can find a subset of vertices $S' \subset V'$ that corresponds 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 $d(S') = |E(S')|/|S'|$. However, the above function will not necessarily lead to subgraphs with high density. Consider the two graphs shown in Figure 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

**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 $G'$.

Figure 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.

Figure 2: Two input graphs (left) and their triangle-graphs (right). The two triangle-graphs are structurally equivalent although the input graphs are not.
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 two algorithms that solve 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 $qs(v)$ the minimum degree of vertex $v$ with respect to the three labels of its adjacent edges in the subgraph induced by $S$, that is $qs(v) = \min_{l \in L(v)} (degS(v,l))$.

4.1 A Supermodular Maximization Approach

In this Section, we 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 $f(S^*) = \max_{S \subseteq V'} f(S)$.

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 $\delta$. What needs to be investigated is what are the properties of the extracted subgraphs and how they differ from the ones extracted from existing methods.

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 $k = 3$. 

**Theorem 1.** Function $h: 2^V \to \mathbb{R}$ defined by $h(S) = \sum_{v \in S} d_S(v) - |S|$ is supermodular.

**Proof.** Let $A, B \subseteq V$ and $v \in V \setminus B$. Let $d: 2^V \to \mathbb{R}$ be a function defined as $d(S) = \sum_{v \in S} \min_{l \in L(v)} (degS(v,l))$. Function $d$ takes as input a set of vertices $S$, computes for each vertex $v \in S$ its degrees with respect to the three labels, and returns the sum of the minimum degrees of the vertices. Given an input set $S \subseteq V$, and two vertices $v \in V \setminus S$ and $u \in \mathbb{R}$. 

**Theorem 2.** Function $h: 2^V \to \mathbb{R}$ is submodular.

**Proof.** Let $A, B \subseteq V$ and $v \in V \setminus B$. Let $d: 2^V \to \mathbb{R}$ be a function defined as $d(S) = \sum_{v \in S} \min_{l \in L(v)} (degS(v,l))$. Function $d$ takes as input a set of vertices $S$, computes for each vertex $v \in S$ its degrees with respect to the three labels, and returns the sum of the minimum degrees of the vertices. Given an input set $S \subseteq V$, and two vertices $v \in V \setminus S$ and $u \in \mathbb{R}$.
S, let \( g_{S,v}(u) \) be a function that measures the incremental value of function \( z(u) \) when \( v \) is added to the input set \( S \), that is \( g_{S,v}(u) = g_{S∪\{u\}}(u) - g_{S}(u) \). Then,

\[
d(B∪\{v\}) − d(B) = \sum_{u∈B∪\{v\}} q_{B∪\{v\}}(u) − \sum_{u∈B} q_{B}(u)
= \sum_{u∈B} g_{B,v}(u) + q_{B∪\{v\}}(v) ≥ \sum_{u∈A} g_{A,v}(u) + q_{A∪\{v\}}(v)
= \sum_{u∈A∪\{v\}} q_{A∪\{v\}}(u) − \sum_{u∈A} q_{A}(u) = d(A∪\{v\}) − d(A)
\]

Since \( B \) is a superset of \( A \), for any vertex \( u ∈ A, B, g_{B,v}(u) ≥ g_{A,v}(u) \). Notice also that \(|B| ≥ |A|\), hence, \( \sum_{u∈A} g_{A,v}(u) ≥ \sum_{u∈A∪\{v\}} q_{A∪\{v\}}(u) \). Finally, \( q_{B∪\{v\}}(v) ≥ q_{A∪\{v\}}(v) \), and therefore, \( d \) is a supermodular function. Furthermore, for any \( α > 0 \) the function \( −α|S| \) is supermodular. Hence, since function \( f \) is the sum of two supermodular functions, it is also supermodular.

To find the set of vertices \( S^* \) that maximizes the triangle hypergraph 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 problem and runs in \( O\left( \frac{m^3}{2} + (t^5(t + y) + t^6) \log t \right) \) 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.

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 \( O(m^{3/2}) \) time \([22]\). As mentioned above, the algorithm will run in a logarithmic number of rounds. Furthermore, Orlin’s algorithm runs in \( O(n^2 EO + n^6) \) time where \( n \) is the size of the ground set and \( EO \) is the time to evaluate \( h(S) \) for some \( S ⊆ V \) \([26]\). 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 \( O\left( \frac{m^3}{2} + (t^5(t + y) + t^6) \log t \right) \).

**Lemma 1.** Algorithm 2 solves the problem and runs in \( O\left( \frac{m^3}{2} + (t^5(t + y) + t^6) \log t \right) \) time.

**Proof.** By solving a series of supermodular maximization problems, Algorithm 2 returns the subgraph \( S^* \) that maximizes the objective function. At each step, the algorithm checks if there exists a set of vertices \( S \) such that the objective value of the induced subgraph is greater or equal to \( α \). To find the set of vertices \( S^* \) that maximizes the objective function, the algorithm performs a binary search on the triangle hypergraph density value \( α \). Specifically, if the value \( val \) returned by the supermodular maximization algorithm is nonnegative, then

\[
h(S) = d(S) − α|S| = val ≥ 0 \Rightarrow \frac{d(S)}{|S|} = f(S) ≥ α
\]

and there exists a set of vertices \( S \) whose triangle-graph density value is greater or equal to \( α \). The set of vertices \( S \) for which the value of the supermodular maximization algorithm is nonnegative, \( val_k ≥ 0 \), and for which the value of the triangle hypergraph density \( α \) is the largest among all sets of vertices corresponds to the optimal set \( S^* \).

As regards the binary search, we set the upper and lower bounds \( u \) and \( l \) as follows. The lower bound is set equal to the triangle hypergraph density of the transformed graph, hence, \( l = \frac{d(V)}{|V|} \). Furthermore, the triangle hypergraph density \( f(S) \) of a set of vertices \( S ⊆ V \) cannot be greater than that of a \(|V|\)-clique where each edge of each vertex has at least \( \delta \) vertices. Hence, we set the upper bound \( u \) equal to \( \frac{|V|}{\frac{|V|}{2} + 1} \). Therefore, the value of the triangle hypergraph density \( f(S^*) \) of the optimal set \( S^* \) will lie between \( l \) and \( u \).

As regards the criterion to stop the binary search, it follows by observing that the smallest distance between two different possible values of the triangle hypergraph density is \( \frac{1}{|V|(|V|−1)} \). The difference \( δ \) between two possible is

\[
δ = \frac{d(S_1)}{|S_1|} − \frac{d(S_2)}{|S_2|} \geq \frac{d(S_1)|S_2| − d(S_2)|S_1|}{|S_1||S_2|}
\]

If \( |S_1| = |S_2| \), then \( |δ| ≥ \frac{1}{|S_1|^2} \geq \frac{1}{|V|^2} \), otherwise \( |δ| ≥ \frac{1}{|S_1||S_2|} ≥ \frac{1}{|V||V|} \). Therefore, if \( |S_1|, S_2 ⊆ V \), \( |δ| ≥ \frac{1}{|V||V|−1} \) \( \forall |V| > 1 \), then for any \( S_1, S_2 ⊆ V \), \( |δ| ≥ \frac{1}{|V||V|−1} \).

### 4.2 A Linear Programming Approach

The problem of computing \( f(S^*) \) can be also expressed as a linear program (LP). Define a variable \( x_{ij} \) for every edge \( ij \in E \) and a variable \( y_i \) for every vertex \( i \in V \). Let also \( l_{ij}^1, l_{ij}^2, l_{ij}^3 \) denote the three possible labels of the edges adjacent to vertex \( y \). The one that is equal to the assigned to \( x_{ij} \) label has a value equal to 1, while the other two are equal to 0. We use the following LP which is inspired by the LP proposed...
by Charikar \cite{Charikar}

\[
\begin{align*}
\text{maximize} \quad & \sum_i z_i \\
\text{subject to} \quad & z_i \leq \sum_j l^1_{ij} x_{ij} \\
& z_i \leq \sum_j l^2_{ij} x_{ij} \\
& z_i \leq \sum_j l^3_{ij} x_{ij} \\
\forall ij \in E, \quad & x_{ij} \leq y_i \\
\forall ij \in E, \quad & x_{ij} \leq y_j \\
& \sum_i y_i \leq 1 \\
& x_{ij}, y_i, z_i \geq 0
\end{align*}
\]

(1)

This corresponds to the objective value of the solution of the LP. Suppose that there exists no value \( r \) such that \( Z(r) / |S(r)| \geq \text{val.} \) Then, we obtain the following contradiction

\[
\text{val} = \int_0^\infty \phi(r)dr < \text{val} \int_0^\infty |S(r)|dr \leq \text{val}
\]

To find a set of vertices \( S \subseteq V \) such that \( f(S) \geq \text{val,} \) we can check all the sets \( S(r) \) obtained by setting \( r \) equal to the different values of \( \bar{y}_i \) for \( i \in V \).

**THEOREM 3.** If the value of the optimum solution of LP \( \mathcal{l} \) is \( \text{OPT}(LP) \), then

\[
f(S^*) = \text{OPT}(LP)
\]

Further, a subset of vertices can be computed from that solution of LP \( \mathcal{l} \) that corresponds to the optimal solution to the problem.

**PROOF.** From Lemma 1, \( f(S^*) \leq \text{OPT}(LP) \). From Lemma 2, \( f(S^*) \geq \text{OPT}(LP) \). Putting the two Lemmas together, we get that \( f(S^*) = \text{OPT}(LP) \) and Lemma 2 provides a constructive procedure for obtaining the set of vertices that maximizes \( f \). \( \square \)

### 4.3 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 hypergraph density \( f(S) \). The proposed algorithm is an adaptation of the greedy algorithm of Asahiro et al. \cite{Asahiro}. The algorithm is illustrated as Algorithm 3. The algorithm iter-

\[
\text{Algorithm 3 Greedy algorithm}
\]

**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 \( S \).

\[
S_{i-1} \leftarrow S_i \setminus \{v\}
\]

end for

\[
S \leftarrow \text{arg max}_{i=1,\ldots,|V|} f(S_i)
\]

atively removes the vertex \( v \) whose value \( d(v) \) is the smallest among all vertices. Subsequently, it computes the triangle hypergraph density of the subgraph induced by the remaining vertices. The output is the subgraph over all the produced subgraphs that maximizes triangle hypergraph density. The algorithm is linear to the number of vertices and the number of edges of the triangle-graph, hence its complexity is \( O(t + y) \) where \( t \) is the number of triangles in the input graph and \( y \) is the number of edges of the triangle-graph.

**THEOREM 4.** 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 \( \text{q}_{S_i}(u) \) be the minimum degree of vertex \( u \) in \( S_i \) with respect to the three labels of its adjacent edges. Then, it holds that

\[
f(S) \geq \frac{|S^*|}{|S|} f_G + \left(1 - \frac{|S^*|}{|S|}\right) \text{q}_{S_i}(u)
\]
The algorithm returns a set of nodes $S$ and a vertex $v$, let $q_S(v)$ be the minimum degree of vertex $v$ with respect to its three labels. Let also $S^*$ be the vertices of the optimal subgraph. The optimal value of the function is obtained for the set of vertices $S^*$ and is equal to $f(S^*) = d(S^*)/|S^*|$. Consider the iteration of the greedy algorithm just before the first vertex $u$ that belongs in the optimal set $S^*$ is removed. Let $S_i$ denote the set of vertices still present before the removal of $u$. The value of the function for the set of vertices $S_i$ is then $f(S_i) = d(S_i)/|S_i|$. Since $S^* \subseteq S_i \subseteq V$, it holds that $q_{S_i}(v) \geq q_{S^*}(v), \forall v$. In each iteration, the algorithm removes the vertex with the minimum degree with respect to the three labels of its adjacent edges. Since $u$ is the first vertex to be removed by the algorithm, it is also easy to see that $q_{S_i}(v) = q_{S_i}(u) \geq q_{S^*}(u)$. Therefore, 

$$ f(S_i) = \frac{d(S_i)}{|S_i|} \geq \frac{\sum_{v \in S^*} q_{S_i}(v) + \sum_{v \in S_i \setminus S^*} (q_{S_i}(v) - q_{S^*}(v)) + \sum_{v \in S_i \setminus S^*} q_{S_i}(v)}{|S_i|} \geq \frac{\sum_{v \in S^*} q_{S_i}(v)}{|S_i|} \geq \frac{|S^* f(S^*) + (|S_i| - |S^*|)q_{S_i}(u)}{|S_i|} = \frac{|S^*| f(S^*) + (1 - \frac{|S^*|}{|S_i|})q_{S_i}(u)}{\frac{|S_i|}{|S_i|}} + \left(1 - \frac{|S^*|}{|S_i|}\right) q_{S_i}(u) $$ 

The algorithm returns a set of nodes $S$ which is the best over all iterations, hence we obtain 

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

\square

From the above result, we can see that the bound provided by the approximation algorithm highly depends on the relationship between $|S_i|$ and 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.

Table 1: Graphs used for evaluating the algorithms.

| Graph       | | |
|-------------|---|---|
| Karate      | 34 | 78 |
| Dolphins    | 62 | 159 |
| Lesmis      | 77 | 254 |
| Adjnoun     | 112 | 425 |
| Football    | 115 | 613 |
| Polbooks    | 105 | 441 |
| Cellegananaeu | 297 | 2,148 |
| Polblogs    | 1,224 | 16,715 |
| Power1      | 4,941 | 6,594 |
| ca-HepTh    | 9,875 | 25,973 |
| Wiki-Vote   | 7,115 | 100,762 |
| ca-CondMat  | 23,133 | 93,439 |
| p2p-Gantea31 | 62,586 | 147,892 |
| Slashdot0902 | 82,168 | 504,230 |
| email-EnAll | 265,009 | 364,481 |
| Amazon      | 334,863 | 925,872 |
| roadNet-PA  | 1,088,092 | 1,541,898 |
| roadNet-CA  | 1,965,206 | 2,766,607 |

5.2 Results and Discussion

Table 1 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 $\delta$ and $\tau$ 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
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 $\frac{1}{2}$ 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 |
|------------|----------|-----------|-----------|------------|------------|------------|------------|
|            | $|S|$ | $\delta$ | $\tau$ | $|S|$ | $\delta$ | $\tau$ | $|S|$ | $\delta$ | $\tau$ | $|S|$ | $\delta$ | $\tau$ | $|S|$ | $\delta$ | $\tau$ |
| 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 | 6 0.86 0.56 | 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 |

Table 3: Comparison of the extracted subgraphs by Charikar’s $\frac{1}{2}$ approximation algorithm for the DS problem (Greedy DS), Tsourakakis’s $\frac{1}{2}$ 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 |
|------------|-----------|------------|------------|------------|
|            | $|S|$ | $\delta$ | $\tau$ | $|S|$ | $\delta$ | $\tau$ | $|S|$ | $\delta$ | $\tau$ |
| 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 |
| ca-HepTh    | 32 1.0 1.0 | 32 1.0 1.0 | 32 1.0 1.0 | 32 1.0 1.0 |
| Wiki-Vote   | 828 0.11 0.004 | 461 0.19 0.014 | 133 0.47 0.131 | 152 0.42 0.104 |
| ca-CondMat  | 26 1.000 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 0.08 | 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 |
| Amazon      | 9 0.91 0.761 | 16 0.45 0.178 | 9 0.91 0.761 | 170 0.03 0.001 |
| roadNet-PA  | 11.286 0.0002 0.0 | 84 0.036 0.0008 | 5 0.80 0.40 | 84 0.036 0.0008 |
| roadNet-CA  | 19.899 0.0001 0.0 | 168 0.017 0.0002 | 5 0.80 0.40 | 168 0.017 0.0002 |

Table 4: Triangle-graph densities of the subgraphs extracted by the exact and the greedy approximation algorithms.

| Dataset    | Exact TGDS | Greedy TGDS |
|------------|------------|------------|
|            | $|S|$ | $\delta$ | $\tau$ | $|S|$ | $\delta$ | $\tau$ |
| Karate     | 2.25 | 2.25 | 2.25 |
| Dolphins   | 2.25 | 2.25 | 2.25 |
| Lesmis     | 7.60 | 7.60 | 7.60 |
| Adjnoun    | 2.39 | 2.36 | 2.36 |
| Football   | 6.0 | 6.0 | 6.0 |
| Polbooks   | 4.02 | 3.89 | 3.89 |

6. 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, and we project the extracted subgraphs back to the input graph. We propose two algorithms 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 for extracting dense subgraphs. 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.
7. REFERENCES

[1] J. I. Alvarez-Hamelin, L. Dall’Asta, A. Barrat, and A. Vespignani. Large scale networks fingerprinting and visualization using the k-core decomposition. In Advances in Neural Information Processing Systems, pages 41–50, 2005.

[2] R. Andersen and K. Chellapilla. Finding Dense Subgraphs with Size Bounds. In Algorithms and Models for the Web-Graph, pages 25–37. Springer, 2009.

[3] A. Angel, N. Koudas, N. Sarkas, D. Srivastava, M. Svendsen, and S. Tirthapura. Dense subgraph maintenance under streaming edge weight updates for real-time story identification. The VLDB Journal, 23(2):175–199, 2014.

[4] Y. Asahiro, R. Hassin, and K. Iwama. Complexity of Finding Dense Subgraphs. Discrete Applied Mathematics, 121(1):15–26, 2002.

[5] Y. Asahiro, K. Iwama, H. Tamaki, and T. Tokuyama. Greedily Finding a Dense Subgraph. Journal of Algorithms, 34(2):203–221, 2000.

[6] G. D. Bader and C. W. Hogue. An automated method for finding molecular complexes in large protein interaction networks. BMC bioinformatics, 4(1):1, 2003.

[7] O. D. Balalau, F. Bonchi, T. Chan, F. Gullo, and M. Sozio. Finding Subgraphs with Maximum Total Density and Limited Overlap. In Proceedings of the 8th International Conference on Web Search and Data Mining, pages 379–388, 2015.

[8] A. Björklund, R. Pagh, V. V. Williams, and U. Zwick. Listing Triangles. In International Colloquium on Automata, Languages, and Programming, pages 223–234, 2014.

[9] C. Bron and J. Kerbosch. Algorithm 457: finding all cliques of an undirected graph. Communications of the ACM, 16(9):575–577, 1973.

[10] G. Buehrer and K. Chellapilla. A Scalable Pattern Mining Approach to Web Graph Compression with Communities. In Proceedings of the 2008 International Conference on Web Search and Data Mining, pages 95–106, 2008.

[11] M. Charikar. Greedy Approximation Algorithms for Finding Dense Components in a Graph. In Approximation Algorithms for Combinatorial Optimization, pages 84–95. 2000.

[12] J. Chen and Y. Saad. Dense Subgraph Extraction with Application to Community Detection. IEEE Transactions on Knowledge and Data Engineering, 24(7):1216–1230, 2012.

[13] N. Chiba and T. Nishizeki. Arboricity and Subgraph Listing Algorithms. SIAM Journal on Computing, 14(1):210–223, 1985.

[14] X. Du, R. Jin, L. Ding, V. E. Lee, and J. H. Thornton Jr. Migration Motif: A Spatial-Temporal Pattern Mining Approach for Financial Markets. In Proceedings of the 15th International Conference on Knowledge Discovery and Data Mining, pages 1135–1144, 2009.

[15] U. Feige. Approximating maximum clique by removing subgraphs. SIAM Journal on Discrete Mathematics, 18(2):219–225, 2004.

[16] U. Feige, D. Peleg, and G. Kortsarz. The Dense k-Subgraph Problem. Algorithmica, 29(3):410–421, 2001.

[17] E. Fratkin, B. T. Naughton, D. L. Brutlag, and S. Batzoglou. Motifcut: regulatory motifs finding with maximum density subgraphs. Bioinformatics, 22(14):e150–e157, 2006.

[18] D. Gibson, R. Kumar, and A. Tomkins. Discovering Large Dense Subgraphs in Massive Graphs. In Proceedings of the 31st International Conference on Very Large Data Bases, pages 721–732, 2005.

[19] A. V. Goldberg. Finding a Maximum Density Subgraph. University of California Berkeley, Technical Report, 1984.

[20] M. Grötschel, L. Lovász, and A. Schrijver. Geometric Algorithms and Combinatorial Optimization, volume 2. Springer Science & Business Media, 2012.

[21] J. Hästad. Clique is hard to approximate within n1−ε. In Proceedings of the 37th Annual Symposium on Foundations of Computer Science, pages 627–636, 1996.

[22] A. Itai and M. Rodeh. Finding a Minimum Circuit in a Graph. SIAM Journal on Computing, 7(4):413–423, 1978.

[23] R. M. Karp. Reducibility among combinatorial problems. Springer, 1972.

[24] S. Khuller and B. Saha. On Finding Dense Subgraphs. In Automata, Languages and Programming, pages 597–608. Springer, 2009.

[25] V. E. Lee, N. Ruan, R. Jin, and C. Aggarwal. A survey of algorithms for dense subgraph discovery. In Managing and Mining Graph Data, pages 303–336. Springer, 2010.

[26] J. B. Orlin. A faster strongly polynomial time algorithm for submodular function minimization. Mathematical Programming, 118(2):237–251, 2009.

[27] T. Schank and D. Wagner. Finding, Counting and Listing all Triangles in Large Graphs, an Experimental Study. In International Workshop on Experimental and Efficient Algorithms, pages 606–609, 2005.

[28] A. Schrijver. A Combinatorial Algorithm Minimizing Submodular Functions in Strongly Polynomial Time. Journal of Combinatorial Theory, Series B, 80(2):346–355, 2000.

[29] M. Sozio and A. Gionis. The Community-search Problem and How to Plan a Successful Cocktail Party. In Proceedings of the 16th International Conference on Knowledge Discovery and Data Mining, pages 939–948, 2010.

[30] C. Tsourakakis. The K-Clique Densest Subgraph Problem. In Proceedings of the 24th International Conference on World Wide Web, pages 1122–1132, 2015.

[31] C. Tsourakakis, F. Bonchi, A. Gionis, F. Gullo, and M. Tsiarli. Denser than the Densest Subgraph: Extracting Optimal Quasi-Cliques with Quality Guarantees. In Proceedings of the 19th International Conference on Knowledge Discovery and Data Mining, pages 104–112, 2013.

[32] N. Wang, J. Zhang, K.-L. Tan, and A. K. Tung. On Triangulation-based Dense Neighborhood Graph
Discovery. Proceedings of the VLDB Endowment, 4(2):58–68, 2010.