

Computing the k Densest Subgraphs of a Graph

Riccardo Dondi

Università degli Studi di Bergamo, Bergamo, Italy
riccardo.dondi@unibg.it

Danny Hermelin

Ben-Gurion University of the Negev, Be'er Sheva, Israel
hermelin@bgu.ac.il

Abstract

Computing cohesive subgraphs is a central problem in graph theory. While many formulations of cohesive subgraphs lead to NP-hard problems, finding a densest subgraph can be done in polynomial-time. As such, the densest subgraph model has emerged as the most popular notion of cohesiveness. Recently, the data mining community has started looking into the problem of computing k densest subgraphs in a given graph, rather than one. In this paper we consider a natural variant of the k densest subgraphs problem, where overlap between solution subgraphs is allowed with no constraint. We show that the problem is fixed-parameter tractable with respect to k , and admits a PTAS for constant k . Both these algorithms complement nicely the previously known $O(n^k)$ algorithm for the problem.

2012 ACM Subject Classification Theory of computation → Graph algorithms analysis; Mathematics of computing → Graph theory; Networks → Network algorithms

Keywords and phrases Algorithm Design, Network Mining and Analysis, Densest Subgraph, Algorithmic Aspects of Networks.

Digital Object Identifier 10.4230/LIPIcs...

1 Introduction

Finding cohesive subgraphs is a central problem in the analysis of social networks [20], graph-mining [26, 28, 29, 27], group dynamics research [8], computational biology [10], and many other areas. The most basic and natural attempt at modeling cohesiveness is via the notion of cliques; however, this notion is too strict and rigid for most applications, and is also known to be computationally hard [15, 32].

While there are several alternative definitions for cohesiveness [18], a notion that has emerged as arguably the most popular is the *densest subgraph* model [1, 5, 11, 23, 26, 28, 31]. Here, the *density* of a graph is simply the edge-to-vertex ratio in the graph, and the densest subgraph is the (induced) subgraph that maximizes this ratio. As opposed to the maximum clique, finding a densest subgraph in a graph is polynomial-time solvable [12, 13, 16, 25]. This fact, along with the naturality of the concept, has lead the notion of density to nowadays be considered at the core of large scale data mining [4].

Recent contributions have shifted the interest from computing a single cohesive subgraph to computing a set of such subgraphs [5, 11, 23, 30], as this is naturally more desirable in most applications. The proposed approaches may allow (but not force) the subgraphs to overlap, as many real-world cohesive groups share common elements. For example, hubs may belong to more than one community [21, 11]. The way the overlap is restricted, if at all, varies among the different approaches. For instance, in [5], the notion of overlap is restricted via a constraint on the pairwise Jaccard coefficient of the subgraphs of the solution, while in [11] the total overlap is factored into the objective function.



© ;

licensed under Creative Commons License CC-BY

Leibniz International Proceedings in Informatics



LIPICS

Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1.1 A natural variant

In this paper we consider a variant of the problem of computing k densest subgraphs of a given graph, where subgraphs in the solution must be distinct (*i.e.* have different vertex sets). Thus a solution subgraph may be a subgraph, a supergraph, or have almost the same vertex set as another solution subgraph. The objective function is the maximization of the total sum of densities of the solution subgraphs.

► **Problem 1.** k-Densest Subgraphs

Input: A graph G .

Output: A set of k pairwise distinct subgraphs G_1, \dots, G_k of G .

Objective: Maximize $\sum_{i=1}^k \text{density}(G_i)$.

While k-Densest Subgraphs is arguably the most basic variant for the problem of computing the k densest subgraphs of a given graph, very little is known about the problem from a theoretical perspective. In [7], it is shown that this problem is solvable in $n^{O(k)}$ time. This is the main yardstick by which we assess the results in this paper.

► **Theorem 1** ([7]). k-Densest Subgraphs can be solved in $n^{O(k)}$ time.

Our first result shows that there is a rather efficient algorithm (for constant values of k), if one is willing to slightly compromise the quality of the solution. In particular, we show that the problem admits an efficient PTAS (EPTAS):

► **Theorem 2.** For any fixed $k, \varepsilon \geq 1$, there is an algorithm that computes in $O(mn \log n)$ time a $(1 - \frac{1}{\varepsilon})$ -approximate solution for k-Densest Subgraphs.

Our second result shows that k-Densest Subgraphs is in fact fixed-parameter tractable when parameterized by the number k of subgraphs. In particular, our second algorithm shows that the problem is polynomial-time solvable even for $k = \Theta(\lg n)$. More precisely, we prove the following:

► **Theorem 3.** k-Densest Subgraphs can be solved in $O(2^k mn^3 \log n)$ time.

1.2 Related work

The Densest Subgraph problem, the problem of computing a densest subgraph in a given graph, is the special case of k-Densest Subgraphs when $k = 1$. This problem has been extensively studied in the literature, and we outline here only the main results. The problem is known to be polynomial-time solvable [13, 25, 12, 16], and it can be approximated within a factor of $\frac{1}{2}$ in linear time [19, 3, 6]. Generalization of the problem to weighted graphs [13], as well as directed graphs [17], also turn out to be polynomial-time solvable. However, the Densest Subgraph problem becomes NP-hard when constraints on the number of vertices in the output graph are added [1, 2, 9, 14, 17, 22].

2 Preliminaries

All graphs considered in this paper are simple, undirected, and without self-loops. Throughout the paper we let $G = (V, E)$ denote an input graph, and we let $n = |V|$ and $m = |E|$. For a vertex $v \in V$, we let $\deg(v)$ denote the degree of v in G , *i.e.* $\deg(v) = |\{u \in V : \{u, v\} \in E\}|$. The density of G is defined by $\text{density}(G) = m/n$, and in general, the density of a graph is the ratio between the number of edges and the number of vertices in the graph.

Given a subset of vertices $V_1 \subseteq V$, we denote by $G[V_1]$ the *subgraph* of G induced by V_1 ; formally, $G[V_1] = (V_1, E_1)$ where $E_1 = \{\{u, v\} \in E : u, v \in V_1\}$. Thus, a subgraph of G is determined completely by its subset of vertices. If $G[V_1]$ and $G[V_2]$ are both subgraphs of G , then we say that these subgraphs are *distinct* whenever $V_1 \neq V_2$. If $V_1 \cap V_2 = \emptyset$ then the two subgraphs are *disjoint*, and if $V_1 \subset V_2$, then $G[V_2]$ is a *proper supergraph* of $G[V_1]$.

2.1 Goldberg's algorithm

As mentioned above, the Densest Subgraph problem can be solved in polynomial-time [13, 25, 12]. The main idea is to reduce the problem to a series of min-cut computations. Picard and Queyranne's algorithm [25] requires $O(n)$ such computations, where n is the number of vertices in the input graph, while Goldberg's algorithm [13] improves this to $O(\log n)$, thus giving an overall time complexity of $O(mn \log n)$ via Orlin's algorithm [24]. Recently, the time complexity of Goldberg's algorithm for unweighted graphs has been improved to $O(n^3)$ [16]. Goldberg also showed that one can compute in $O(mn \log n)$ time a densest subgraph in a *vertex-weighted* graph; here, the density of a vertex-weighted graph H on n vertices of total weight w and m edges is given by $\text{density}(H) = (m + w)/n$.

3 An EPTAS for k-Densest Subgraphs

In the following section we describe our EPTAS for k-Densest Subgraphs. Let (G, k) denote a given instance of k-Densest Subgraphs, and let $\varepsilon > 0$ be a given constant. Our goal is to compute in $O(mn \log n)$ time k distinct subgraphs G_1, \dots, G_k of G with densities d_1, \dots, d_k such that $\sum_i d_i \geq (1 - \frac{1}{\varepsilon}) \cdot OPT$, where OPT is the value of an solution of k-Densest Subgraphs, that is the total sum of densities of the k densest subgraphs in G . Recall that $k = O(1)$.

Below we first provide a description of our algorithm, followed by an analysis of its running time, and an analysis of its approximation ratio guarantee. Since the function $(\frac{n-2k}{n})^k$ tends to 1 as n grows to infinity, we will henceforth assume that n is sufficiently large so that the following inequality holds (otherwise we can solve the problem optimally via brute force in $O(1)$ time):

$$\left(\frac{n-2k}{n}\right)^k \geq \left(1 - \frac{1}{\varepsilon}\right). \quad (1)$$

3.1 The algorithm

We say that a subgraph $G_i = (V_i, E_i)$ of G is *small* if $|V_i| \leq \varepsilon - 1$. Our algorithm proceeds in a certain way so long that all subgraphs computed so far are small; once a subgraph which is not small is computed, the algorithm proceeds in a different manner. The first subgraph $G_1 = (V_1, E_1)$ is computed using Goldberg's algorithm, so G_1 is a densest subgraph in G .

Suppose that we have computed subgraphs G_1, \dots, G_i for some $1 \leq i \leq k-1$, and all these subgraphs are small. The subgraph G_{i+1} is taken to be a densest graph out of all of the following possible candidates:

- A densest subgraph in $G[V \setminus \{v_1, \dots, v_i\}]$ for some $v_1 \in V_1, \dots, v_i \in V_i$.
- A densest strict supergraph of G_j in G for some $j \in \{1, \dots, i\}$.

Note that some of the candidates of the second type above can be graphs in $\{G_1, \dots, G_i\}$; such graphs are naturally excluded from being candidates for the subgraph G_{i+1} .

Suppose that we have computed subgraphs G_1, \dots, G_i for some $1 \leq i \leq k-1$, and $G_i = (V_i, E_i)$ is not small. Then in this case G_i can either be big or huge. We say that

XX:4 Computing the k Densest Subgraphs of a Graph

G_i is *big* if $|V_i| \leq n - k - i$, and otherwise it is *huge*. If G_i is big, we choose arbitrary distinct vertices $v_{i+1}, \dots, v_k \in V \setminus V_i$ and set G_j to be the graph induced by $V_i \cup \{v_j\}$ for $j \in \{i+1, \dots, k\}$. Note that since V_i is not huge, there are enough distinct vertices in $V \setminus V_i$. Also note that as G_i is the only big subgraph in G_1, \dots, G_i , it is not a proper subgraph of any of these graphs and so all subgraphs G_j are distinct from all subgraphs computed so far.

If G_i is huge, then the graphs G_{i+1}, \dots, G_k are computed by iteratively removing minimal degree vertices in G_i . Since G_i is huge and all graphs G_1, \dots, G_{i-1} are small, we are guaranteed that subgraphs computed in this way are distinct from those we have computed.

3.2 Run-time analysis

Before analyzing the run-time of our algorithm, we begin with the following lemma:

► **Lemma 4.** *Let H_0 be a strict subgraph of G , and let H be a densest strict supergraph of H_0 in G . If $\text{density}(H) \leq \text{density}(H_0)$, then there is an algorithm that computes in $O(mn \log n)$ time a strict supergraph of H_0 in G with density equal to $\text{density}(H)$, given H_0 as input.*

Proof. Given $H_0 = (V_0, E_0)$ as input, the algorithm uses Goldberg's algorithm to compute a densest subgraph $H_1 = (V_1, E_1)$ in the vertex-weighted graph $G^* = G[V \setminus V_0]$, with vertex weights defined by $w(v) = |N_G(v) \cap V_0|$ for each vertex v of G^* . It then returns the graph $H = H_0 \cup H_1 = G[V_0 \cup V_1]$ as a solution. Clearly, this can be done in $O(mn \log n)$ time, and H is a strict supergraph of H_0 in G . We claim that H is indeed a densest among all supergraphs of H_0 .

Let $H' = (V', E')$ be any strict supergraph of H_0 ($V_0 \subset V'$), and let $H_2 = (V_2, E_2)$ be the subgraph of G induced by $V_2 = V' \setminus V_0$. Our goal is to show that H is at least as dense as H' in G . Let $n_i = |V_i|$ and $m_i = |E_i| + \sum_{v \in H_i} w(v)$ for $i \in \{1, 2\}$. Then the density of H_1 and H_2 in the vertex weighted graph G^* is $d_1 = m_1/n_1$ and $d_2 = m_2/n_2$ respectively. Also, by letting $n_0 = |V_0|$ and $m_0 = |E_0|$, the density of H_0 in G is given by $\text{density}(H_0) = d_0 = m_0/n_0$. Furthermore, observe that by the definition of the vertex weight function in G^* , we have

$$\begin{aligned} \text{density}(H) &= \frac{|E_0| + |E_1| + |E(V_0, V_1)|}{|V_0| + |V_1|} = \frac{|E_0| + |E_1| + \sum_{v \in V_1} |N(v) \cap V_0|}{|V_0| + |V_1|} = \\ &= \frac{|E_0| + |E_1| + \sum_{v \in V_1} w(v)}{|V_0| + |V_1|} = \frac{m_0 + m_1}{n_0 + n_1}, \end{aligned}$$

and similarly, $\text{density}(H') = (m_0 + m_2)/(n_0 + n_2)$. Below we argue that $\text{density}(H)$ is at least as large as $\text{density}(H')$.

By standard algebra, we have

$$\begin{aligned} \text{density}(H) \geq \text{density}(H') &\iff \\ \frac{m_0 + m_1}{n_0 + n_1} \geq \frac{m_0 + m_2}{n_0 + n_2} &\iff \\ m_0 n_2 + m_1 (n_0 + n_2) \geq m_0 n_1 + m_2 (n_0 + n_1) &\iff \\ m_1 n_2 + m_0 (n_2 - n_1) \geq m_2 n_1 + n_0 (m_2 - m_1). & \end{aligned}$$

Thus, to complete the proof it suffices to prove the following two inequalities: $m_1 n_2 \geq m_2 n_1$ and $m_0 (n_2 - n_1) \geq n_0 (m_2 - m_1)$.

For the first inequality, observe that $d_1 = m_1/n_1 \geq d_2 = m_2/n_2$ as H_1 is a densest subgraph in G^* ; this directly implies $m_1 n_2 \geq m_2 n_1$. For second inequality, by the assumption

that $\text{density}(H_0) \geq \text{density}(H)$, we have:

$$\begin{aligned}
\text{density}(H_0) \geq \text{density}(H) &\iff \\
\frac{m_0}{n_0} \geq \frac{m_0 + m_1}{n_0 + n_1} &\iff \\
m_0 n_1 \geq m_1 n_0 &\iff \\
\frac{m_0}{n_0} \geq \frac{m_1}{n_1} &\iff \\
\text{density}(H_0) \geq d_1,
\end{aligned}$$

Thus,

$$\frac{m_0}{n_0} = \text{density}(H_0) \geq d_1 = \frac{d_1(n_2 - n_1)}{n_2 - n_1} \geq \frac{d_2 n_2 - d_1 n_1}{n_2 - n_1} = \frac{m_2 - m_1}{n_2 - n_1},$$

and so the second inequality also holds. \blacktriangleleft

Now, first observe that G_1 is computed in $O(mn \log n)$ time (or $O(n^3)$ time if $m \log n > n^2$) with Goldberg's algorithm given in [13, 16]. Next, note that if some subgraph G_i is big or huge, then the remaining graphs G_{i+1}, \dots, G_k can easily be computed in $O(m + n)$ time. Consider then a small subgraph G_i for some $i \leq k - 1$. Then, by construction, all subgraphs G_1, \dots, G_i are small, and so we have $|V_1| \dots |V_i| \leq \varepsilon^k = O(1)$. The subgraph G_{i+1} is computed by first computing candidates of two different types. For the first type we need to invoke Goldberg's algorithm on a graph $|V_1| \dots |V_i| \leq \varepsilon^k = O(1)$ times, so this requires $O(mn \log n)$ time (or $O(n^3)$ time if $m \log n > n^2$). For the second type, we need to invoke Goldberg's algorithm on a weighted graph, as described in Lemma 4 above, $i = O(1)$ times, and so this also requires $O(mn \log n)$ time. In total, we compute each subgraph G_i in $O(mn \log n)$ time, which gives the same run-time for the entire algorithm since $k = O(1)$.

3.3 Approximation-ratio analysis

Let G_1^*, \dots, G_k^* be an optimal solution of Densest Subgraph on instance G , with densities $d_1^* \geq d_2^* \geq \dots \geq d_k^*$. We analyze the approximation ratio guaranteed by our algorithm by comparing the density of each subgraph $G_i = (V_i, E_i)$ computed by the algorithm with d_i^* . For G_1 this is easy. Since G_1^* is a densest subgraph in G , and G_1 is the graph computed by Goldberg's algorithm, we have:

► **Lemma 5.** $\text{density}(G_1) = d_1^*$.

For the remaining graphs, our analysis splits into three cases depending on the type of graph previously computed by the algorithm.

► **Lemma 6.** If G_i is small, for $i < k$, then $\text{density}(G_{i+1}) = d_{i+1}^*$.

Proof. The optimal subgraph $G_{i+1}^* = (V_{i+1}^*, E_{i+1}^*)$ is either a supergraph of some graph in G_1, \dots, G_i , or $V_j \setminus V_{i+1}^* \neq \emptyset$ for each $j \in \{1, \dots, i\}$. Since the candidates for G_{i+1} considered by our algorithm in case G_i is small cover both these cases, the lemma follows. \blacktriangleleft

Note that Lemma 5 and Lemma 6 together imply that if all subgraphs computed by the algorithm are small, then $\text{density}(G_i) = d_i^*$ for each $i \in \{1, \dots, k\}$, and our algorithm computes an optimal solution. Furthermore, the first big or huge subgraph it computes also has optimal densities. The next two lemmas deal with the remaining subgraphs that are computed after computing a big or huge subgraph.

XX:6 Computing the k Densest Subgraphs of a Graph

► **Lemma 7.** Suppose G_i , for $i < k$, is the first big subgraph computed by the algorithm. Then $\text{density}(G_j) \geq (1 - \frac{1}{\varepsilon}) \cdot d_j^*$ for each $j \in \{i+1, \dots, k\}$.

Proof. Let $n_i = |V_i|$ and $m_i = |E_i|$. By Lemma 5 and Lemma 6 we know that $m_i/n_i = d_i^*$. Furthermore, as G_i is big, we have $n_i > \varepsilon - 1$, or written differently $n_i/(\varepsilon - 1) > 1$. Now, as each G_j has $n_i + 1$ vertices and at least m_i edges, we have

$$\text{density}(G_j) \geq \frac{m_i}{n_i + 1} > \frac{m_i}{n_i + n_i/(\varepsilon - 1)} = \frac{\varepsilon - 1}{\varepsilon} \cdot \frac{m_i}{n_i} = (1 - \frac{1}{\varepsilon}) \cdot d_i^* \geq (1 - \frac{1}{\varepsilon}) \cdot d_j^*. \quad \blacktriangleleft$$

► **Lemma 8.** Suppose G_i , for $i < k$, is the first huge subgraph computed by the algorithm. Then $\text{density}(G_j) \geq (1 - \frac{1}{\varepsilon}) \cdot d_j^*$ for each $j \in \{i+1, \dots, k\}$.

Proof. Let $n_i = |V_i|$ and $m_i = |E_i|$. Since G_i is huge we know that $n_i > n - k$, and again by Lemmas 5 and 6 we know that $m_i/n_i = d_i^*$. Let $v \in V_i$ be a vertex of minimum degree in G_i . Consider the subgraph G_{i+1} , constructed from G_i by removing the vertex $v \in V_i$ with minimum degree. Then the degree of v cannot exceed the average degree in G_i , and so $\deg(v) \leq 2m_i/n_i$. Thus, the density of G_i can be bounded by:

$$\text{density}(G_{i+1}) = \frac{m_i - \deg(v)}{n_i - 1} \geq \frac{m_i - 2m_i/n_i}{n_i - 1} = \frac{n_i - 2}{n_i - 1} \cdot d_i^* > \frac{n - k - 2}{n} \cdot d_i^*.$$

Extending this argument, it can be seen that the density of G_{i+j} , for any $j \in \{1, \dots, k - i\}$, is bounded from below by $\left(\frac{n-k-j-1}{n}\right)^j \cdot d_i^*$. The lemma then directly follows from Equation 1. ◀

Summarizing, due to Lemmas 5, 6, 7, and 8, we know that $\text{density}(G_i) \geq (1 - \frac{1}{\varepsilon}) \cdot d_i^*$ for all $i \in \{1, \dots, k\}$, and so in total we have: $\sum_{i=1}^k \text{density}(G_i) \geq \sum_{i=1}^k (1 - \frac{1}{\varepsilon}) \cdot d_i^* = (1 - \frac{1}{\varepsilon}) \cdot OPT$. This completes the proof of Theorem 2.

4 k-Densest Subgraphs in FPT Time

We next show that **k-Densest Subgraphs** is solvable in $O(2^k mn^3 \log n)$ time, *i.e.* that it is fixed-parameter tractable in k . Recall that our goal is to compute k subgraphs G_1, \dots, G_k of $G = (V, E)$ whose total density is maximal, and our only constraint is that these subgraphs need to be distinct.

Similarly to Section 3, our approach here is to iteratively compute G_1 , then G_2 , and so forth, where we start from a densest subgraph G_1 of G . In what follows, we assume we have already computed the subgraphs $G_1 = (V_1, E_1), \dots, G_\ell = (V_\ell, E_\ell)$, for $\ell \in \{1, \dots, k-1\}$, and our goal is to compute a densest subgraph $G_{\ell+1} = (V_{\ell+1}, E_{\ell+1})$ among all subgraphs in G distinct from G_1, \dots, G_ℓ . Let $V^* = \bigcup_{i=1}^\ell V_i$. We consider the following two cases:

1. There is some vertex $v \in V_{\ell+1}$ that is not in V^* , *i.e.* $V_{\ell+1} \not\subseteq V^*$.
2. $V_{\ell+1}$ is contained completely in V^* , *i.e.* $V_{\ell+1} \subseteq V^*$.

We compute a densest subgraph in each one of these cases, and then take the densest of the two to be $G_{\ell+1}$.

4.1 First case

The first case where $V_{\ell+1} \not\subseteq V^*$ is easy: we iterate through all vertices $v \in V \setminus V^*$ and compute a densest subgraph of G that includes v , and then take the densest of all these subgraphs (each of them being distinct from G_1, \dots, G_ℓ).

► **Lemma 9.** *Let $v \in V$. A densest subgraph of G that includes v can be computed in $O(mn \log n)$ time.*

Proof. Let $w_v : V \rightarrow \mathbb{N}$ be the weight function defined by $w_v(v) = n^2$, and $w_v(u) = 1$ for all vertices $u \neq v$. Then any subgraph of G that does not include v has weighted density less than n , and any subgraph that includes v has weight density at least n . It follows that computing a densest subgraph of G that includes v can be done by a single application of Goldberg's algorithm in $O(mn \log n)$ time on G weighted by w_v . ◀

► **Lemma 10.** *If $V_{\ell+1} \not\subseteq V^*$ then $G_{\ell+1}$ can be computed in $O(mn^2 \log n)$ time.*

Proof. Iterate on all $O(n)$ vertices $v \in V \setminus V^*$, and run the algorithm in Lemma 9 for each such vertex v . In total, by Lemma 9 this takes $O(n) \cdot O(mn \log n) = O(mn^2 \log n)$ time. ◀

4.2 Second case

The second case where $V_{\ell+1} \subseteq V^*$ requires more details. We say that a non-empty subset $\mathcal{C} \subseteq \{V_1, \dots, V_\ell\}$ covers $V_{\ell+1}$ if $V_{\ell+1} \subseteq V_{\mathcal{C}} = \bigcup_{V_i \in \mathcal{C}} V_i$, and it is a *minimal cover* if $V_{\ell+1} \not\subseteq V_{\mathcal{C}'}$ for any proper subset $\mathcal{C}' \subset \mathcal{C}$. Our approach is to compute for each non-empty subset $\mathcal{C} \subseteq \{V_1, \dots, V_\ell\}$, a densest subgraph of G for which \mathcal{C} is a minimal cover.

► **Lemma 11.** *Let $\mathcal{C} \subseteq \{V_1, \dots, V_\ell\}$, and suppose that \mathcal{C} is a minimal cover of $V_{\ell+1}$. If $V_{\ell+1} \neq V_{\mathcal{C}}$, then there are two vertices $v_{in}, v_{out} \in V_{\mathcal{C}}$ such that $v_{in} \in V_{\ell+1}$ and $v_{out} \notin V_{\ell+1}$, and there is no subset $V_i \in \mathcal{C}$ with $v_{in} \in V_i$ and $v_{out} \notin V_i$.*

Proof. Suppose that $V_{\ell+1} \neq V_{\mathcal{C}}$, and so $V_{\ell+1} \subset V_{\mathcal{C}}$. It follows that there exists a vertex $v_{out} \in V_{\mathcal{C}} \setminus V_{\ell+1}$. Consider the subset $\mathcal{C}' \subset \mathcal{C}$ which includes all vertex subsets in \mathcal{C} that do not include the vertex v_{out} , i.e. $\mathcal{C}' = \{V_i \in \mathcal{C} : v_{out} \notin V_i\}$. Note that \mathcal{C}' is indeed a proper subset of \mathcal{C} , as v_{out} belongs to some graph in \mathcal{C} . If $\mathcal{C}' = \emptyset$, then v_{out} belongs to every subset $V_i \in \mathcal{C}$, and the lemma holds. If $\mathcal{C}' \neq \emptyset$, there must be some vertex $v_{in} \in V_{\ell+1} \setminus V_{\mathcal{C}'}$ by the minimality of \mathcal{C} , since otherwise \mathcal{C}' would cover $V_{\ell+1}$. ◀

► **Lemma 12.** *If $V_{\ell+1} \subseteq V^*$ then G_ℓ can be computed in $O(2^k mn^3 \log n)$ time.*

Proof. We iterate over all possible $2^\ell - 1$ non-empty subsets $\mathcal{C} \subseteq \{V_1, \dots, V_\ell\}$. For each subset \mathcal{C} , we iterate over all $O(n^2)$ vertices $v_{in}, v_{out} \in V_{\mathcal{C}}$ and compute a densest subgraph in $G[V_{\mathcal{C}} \setminus \{v_{out}\}]$ that includes v_{in} (using the algorithm in Lemma 9). This requires $O(mn^3 \log n)$ time in total. Out of all subgraphs computed this way, along with all subgraphs of the form $G[V_{\mathcal{C}}]$, we choose the densest subgraph which is distinct from $\{G_1, \dots, G_\ell\}$. As $G_{\ell+1}$ is a densest subgraph in $G[V_{\mathcal{C}} \setminus \{v_{out}\}]$ that includes v_{in} , for the minimal cover \mathcal{C} of $V_{\ell+1}$ and some $v_{in}, v_{out} \in V_{\mathcal{C}}$ (according to Lemma 11), this algorithm is indeed guaranteed to find a subgraph of G with density at least $\text{density}(G_{\ell+1})$. ◀

4.3 Summary

Thus, taking the densest of the subgraph given by Lemma 10 and the subgraph given by Lemma 12 gives us a densest subgraph in G which is distinct from $\{G_1, \dots, G_\ell\}$ in $O(2^k mn^3 \log n)$ time. In this way, we can compute k densest distinct subgraphs of G in $O(2^k k mn^3 \log n)$ time, completing the proof of Theorem 3.

5 Conclusion

This paper studies a natural variant for computing k densest subgraphs of a given graph, a central problem in graph data mining. We show that the problem is fixed-parameter tractable with respect to k , and admits a PTAS for $k = O(1)$.

From a theoretical perspective, the most interesting problem that is left open by our paper is whether **k -Densest Subgraphs** is NP-hard for unbounded k . However, we feel that for most practical settings, the number k of solution subgraphs should be significantly smaller than the size n of the network. Thus, we feel that examining the problem on specific social network models might be more interesting from a practical point of view. Finally, we have considered unweighted graphs, a natural direction is whether it is possible to extend the results to edge-weighted graphs.

Acknowledgements

We thank an anonymous reviewer for pointing out an error in an algorithm included in a previous version of the paper.

References

- 1 Reid Andersen and Kumar Chellapilla. Finding dense subgraphs with size bounds. In Konstantin Avrachenkov, Debora Donato, and Nelly Litvak, editors, *Algorithms and Models for the Web-Graph, 6th International Workshop, WAW 2009, Barcelona, Spain, February 12-13, 2009. Proceedings*, volume 5427 of *Lecture Notes in Computer Science*, pages 25–37. Springer, 2009. doi:[10.1007/978-3-540-95995-3_3](https://doi.org/10.1007/978-3-540-95995-3_3).
- 2 Yuichi Asahiro, Refael Hassin, and Kazuo Iwama. Complexity of finding dense subgraphs. *Discrete Applied Mathematics*, 121(1-3):15–26, 2002. doi:[10.1016/S0166-218X\(01\)00243-8](https://doi.org/10.1016/S0166-218X(01)00243-8).
- 3 Yuichi Asahiro, Kazuo Iwama, Hisao Tamaki, and Takeshi Tokuyama. Greedily finding a dense subgraph. In Rolf G. Karlsson and Andrzej Lingas, editors, *Algorithm Theory - SWAT '96, 5th Scandinavian Workshop on Algorithm Theory, Reykjavík, Iceland, July 3-5, 1996, Proceedings*, volume 1097 of *Lecture Notes in Computer Science*, pages 136–148. Springer, 1996. doi:[10.1007/3-540-61422-2_127](https://doi.org/10.1007/3-540-61422-2_127).
- 4 Bahman Bahmani, Ravi Kumar, and Sergei Vassilvitskii. Densest subgraph in streaming and mapreduce. *PVLDB*, 5(5):454–465, 2012. doi:[10.14778/2140436.2140442](https://doi.org/10.14778/2140436.2140442).
- 5 Oana Denisa Balalau, Francesco Bonchi, T.-H. Hubert Chan, Francesco Gullo, and Mauro Sozio. Finding subgraphs with maximum total density and limited overlap. In Xueqi Cheng, Hang Li, Evgeniy Gabrilovich, and Jie Tang, editors, *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining, WSDM 2015*, pages 379–388. ACM, 2015. doi:[10.1145/2684822.2685298](https://doi.org/10.1145/2684822.2685298).
- 6 Moses Charikar. Greedy approximation algorithms for finding dense components in a graph. In Klaus Jansen and Samir Khuller, editors, *Approximation Algorithms for Combinatorial Optimization, Third International Workshop, APPROX 2000, Proceedings*, volume 1913 of *Lecture Notes in Computer Science*, pages 84–95. Springer, 2000. doi:[10.1007/3-540-44436-X](https://doi.org/10.1007/3-540-44436-X).
- 7 Riccardo Dondi, Mohammad Mehdi Hosseinzadeh, Giancarlo Mauri, and Italo Zoppis. Top- k overlapping densest subgraphs: approximation algorithms and computational complexity. *J. Comb. Optim.*, 41(1):80–104, 2021. doi:[10.1007/s10878-020-00664-3](https://doi.org/10.1007/s10878-020-00664-3).
- 8 Lata Dyaram and T. J. Kamalanabhan. Unearthed: The other side of group cohesiveness. *Journal of Social Sciences*, 10(3):185–190, 2005.
- 9 Uriel Feige, Guy Kortsarz, and David Peleg. The dense k -subgraph problem. *Algorithmica*, 29(3):410–421, 2001. doi:[10.1007/s004530010050](https://doi.org/10.1007/s004530010050).

- 10 Eugene Fratkin, Brian T. Naughton, Douglas L. Brutlag, and Serafim Batzoglou. Motifcut: regulatory motifs finding with maximum density subgraphs. *Bioinformatics*, 22(14):156–157, 2006. doi:10.1093/bioinformatics/btl243.
- 11 Esther Galbrun, Aristides Gionis, and Nikolaj Tatti. Top-k overlapping densest subgraphs. *Data Min. Knowl. Discov.*, 30(5):1134–1165, 2016. doi:10.1007/s10618-016-0464-z.
- 12 Giorgio Gallo, Michael D. Grigoriadis, and Robert Endre Tarjan. A fast parametric maximum flow algorithm and applications. *SIAM Journal on Computing*, 18(1):30–55, 1989. doi:10.1137/0218003.
- 13 Andrew V. Goldberg. Finding a maximum density subgraph. Technical report, Berkeley, CA, USA, 1984.
- 14 Doron Goldstein and Michael Langberg. The dense k subgraph problem. *CoRR*, abs/0912.5327, 2009. arXiv:0912.5327.
- 15 Richard M. Karp. Reducibility among combinatorial problems. In Raymond E. Miller and James W. Thatcher, editors, *Proceedings of a symposium on the Complexity of Computer Computations*, The IBM Research Symposia Series, pages 85–103. Plenum Press, New York, 1972.
- 16 Yasushi Kawase and Atsushi Miyauchi. The densest subgraph problem with a convex/concave size function. *Algorithmica*, 80(12):3461–3480, 2018. doi:10.1007/s00453-017-0400-7.
- 17 Samir Khuller and Barna Saha. On finding dense subgraphs. In Susanne Albers, Alberto Marchetti-Spaccamela, Yossi Matias, Sotiris E. Nikoletseas, and Wolfgang Thomas, editors, *Automata, Languages and Programming, 36th International Colloquium, ICALP 2009, Rhodes, Greece, July 5-12, 2009, Proceedings, Part I*, volume 5555 of *Lecture Notes in Computer Science*, pages 597–608. Springer, 2009. doi:10.1007/978-3-642-02927-1_50.
- 18 Christian Komusiewicz. Multivariate algorithmics for finding cohesive subnetworks. *Algorithms*, 9(1):21, 2016.
- 19 Guy Kortsarz and David Peleg. Generating sparse 2-spanners. *J. Algorithms*, 17(2):222–236, 1994. doi:10.1006/jagm.1994.1032.
- 20 Ravi Kumar, Prabhakar Raghavan, Sridhar Rajagopalan, and Andrew Tomkins. Trawling the web for emerging cyber-communities. *Computer Networks*, 31(11-16):1481–1493, 1999. doi:10.1016/S1389-1286(99)00040-7.
- 21 Jure Leskovec, Kevin J. Lang, Anirban Dasgupta, and Michael W. Mahoney. Community structure in large networks: Natural cluster sizes and the absence of large well-defined clusters. *Internet Mathematics*, 6(1):29–123, 2009. doi:10.1080/15427951.2009.10129177.
- 22 Pasin Manurangsi. Almost-polynomial ratio eth-hardness of approximating densest k-subgraph. In Hamed Hatami, Pierre McKenzie, and Valerie King, editors, *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017*, pages 954–961. ACM, 2017. doi:10.1145/3055399.3055412.
- 23 Muhammad Anis Uddin Nasir, Aristides Gionis, Gianmarco De Francisci Morales, and Sarunas Girdzijauskas. Fully dynamic algorithm for top- k densest subgraphs. In Ee-Peng Lim, Marianne Winslett, Mark Sanderson, Ada Wai-Chee Fu, Jimeng Sun, J. Shane Culpepper, Eric Lo, Joyce C. Ho, Debora Donato, Rakesh Agrawal, Yu Zheng, Carlos Castillo, Aixin Sun, Vincent S. Tseng, and Chenliang Li, editors, *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, CIKM 2017*, pages 1817–1826. ACM, 2017. doi:10.1145/3132847.3132966.
- 24 James B. Orlin. Max flows in $O(nm)$ time, or better. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*, pages 765–774. ACM, 2013. URL: <https://doi.org/10.1145/2488608.2488705>. doi:10.1145/2488608.2488705.
- 25 Jean-Claude Picard and Maurice Queyranne. A network flow solution to some nonlinear 0-1 programming problems, with applications to graph theory. *Networks*, 12(2):141–159, 1982. URL: <https://doi.org/10.1002/net.3230120206>. doi:10.1002/net.3230120206.

XX:10 Computing the k Densest Subgraphs of a Graph

- 26 Mauro Sozio and Aristides Gionis. The community-search problem and how to plan a successful cocktail party. In Bharat Rao, Balaji Krishnapuram, Andrew Tomkins, and Qiang Yang, editors, *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, July 25-28, 2010*, pages 939–948. ACM, 2010. doi:10.1145/1835804.1835923.
- 27 Nikolaj Tatti. Density-friendly graph decomposition. *ACM Trans. Knowl. Discov. Data*, 13(5):54:1–54:29, 2019. doi:10.1145/3344210.
- 28 Nikolaj Tatti and Aristides Gionis. Density-friendly graph decomposition. In Aldo Gangemi, Stefano Leonardi, and Alessandro Panconesi, editors, *Proceedings of the 24th International Conference on World Wide Web, WWW 2015, Florence, Italy, May 18-22, 2015*, pages 1089–1099. ACM, 2015. doi:10.1145/2736277.2741119.
- 29 Charalampos E. Tsourakakis. The k -clique densest subgraph problem. In Aldo Gangemi, Stefano Leonardi, and Alessandro Panconesi, editors, *Proceedings of the 24th International Conference on World Wide Web, WWW 2015, Florence, Italy, May 18-22, 2015*, pages 1122–1132. ACM, 2015. doi:10.1145/2736277.2741098.
- 30 Elena Valari, Maria Kontaki, and Apostolos N. Papadopoulos. Discovery of top- k dense subgraphs in dynamic graph collections. In Anastasia Ailamaki and Shawn Bowers, editors, *Scientific and Statistical Database Management - 24th International Conference, SSDBM 2012, Chania, Crete, Greece, June 25-27, 2012. Proceedings*, volume 7338 of *Lecture Notes in Computer Science*, pages 213–230. Springer, 2012.
- 31 Zhaonian Zou. Polynomial-time algorithm for finding densest subgraphs in uncertain graphs. In *Proceedings of Internation Workshop on Mining and Learning with Graphs*, 2013.
- 32 David Zuckerman. Linear degree extractors and the inapproximability of max clique and chromatic number. *Theory of Computing*, 3(1):103–128, 2007. doi:10.4086/toc.2007.v003a006.