Exact and superpolynomial approximation algorithms for the densest k-subgraph problem*

Nicolas Bourgeois[(a)] Aristotelis Giannakos[(b)] Giorgio Lucarelli[(c)]
Ioannis Milis[(d)] Vangelis Th. Paschos[(b)]

April 13, 2017

Abstract

The DENSEST k-SUBGRAPH problem is a generalization of the maximum clique problem, in which we are given a graph and a positive integer k, and we search among all the subsets of k vertices of the input graph for the subset which induces the maximum number of edges. DENSEST k-SUBGRAPH is a well known optimization problem with various applications as, for example, in the design of public encryption schemes, the evaluation of certain financial derivatives, the identification of communities with similar characteristics, etc. In this paper, we first present algorithms for finding exact solutions for DENSEST k-SUBGRAPH which improve upon the standard exponential time complexity of an exhaustive enumeration by creating a link between the computation of an optimum for this problem to the computation of other graph-parameters such as dominating set, vertex cover, longest path, etc. An FPT algorithm is also proposed which considers as a parameter the size of the minimum vertex cover. Finally, we present several approximation algorithms which run in moderately exponential or parameterized time, describing trade-offs between complexity and approximability. In contrast with most of the algorithms in the bibliography, our algorithms need only polynomial space.

Keywords— combinatorial optimization; dense subgraphs; exact and parameterized algorithms; superpolynomial approximation algorithms

1 Introduction

In the DENSEST k-SUBGRAPH problem we are given a graph $G = (V, E)$ and an integer $k \in \mathbb{N}^+$, and we ask for a subset $A \subseteq V$ of k vertices such that the number of edges induced by A is maximized. In the decision version of this problem, we are given a graph $G = (V, E)$ and two integers $k \in \mathbb{N}^+$ and $\ell \in \mathbb{N}^+$, and we ask to find k vertices that induce a subgraph with at least ℓ edges.

*An extended abstract of a preliminary version of this paper has been presented in WALCOM'13.

[(a)] nbourgeo@phare.normalesup.org, SAMM, Université Paris I, Paris, France

[(b)] {aristotelis.giannakos,paschos}@lamsade.dauphine.fr, Université Paris-Dauphine, PSL Research University, CNRS, UMR 7243 LAMSADE, 75016 Paris, France

[(c)] giorgio.lucarelli@inria.fr, Univ. Grenoble Alpes, CNRS, INRIA, LIG, F-38000 Grenoble, France (corresponding author)

[(d)] milis@aueb.gr, Athens University of Economics and Business, Department of Informatics, Athens, Greece

The DENSEST k-SUBGRAPH problem belongs to a known class of problems, called *fixed cardinality problems*, most of which are generalizations of well-known combinatorial optimization problems. For instance, this is the case for DENSEST k-SUBGRAPH with respect to the MAX CLIQUE problem. Similarly, the MAX k-COVER problem is a generalization of the MIN VERTEX COVER problem, the SPARSEST k-SUBGRAPH of the MAX INDEPENDENT SET problem, etc. As these latter problems are between the 21 first NP-complete problems in [32], the NP-completeness of the former ones is immediately derived. DENSEST k-SUBGRAPH is closely related to the well-known planted dense subgraph problem, very frequently encountered in the design of public encryption schemes and the evaluation of certain financial derivatives (the reader is referred to [24] for more details).

In this paper, we explore the DENSEST k-SUBGRAPH problem from several algorithmic points of view. Our results are analyzed based on the following well-known definitions. A problem is fixed-parameter tractable (FPT) with respect to a parameter t if it can be solved (to optimality) with time-complexity $O(f(t)\cdot p(|I|))$ where f is a function and p is a polynomial in the size of the instance I. A problem Π parameterized by some parameter t is W[1]-hard, if the MAX CLIQUE problem (parameterized by the solution size) reduces to Π by a *fixed-parameter tractable reduction* [22]. W[1]-hard problems do not admit fixed-parameter algorithms, under the assumption that $FPT \neq \text{W[1]}$. A problem parameterized by t belongs to the class XP if it can be solved in polynomial time for any fixed value of t, i.e., by an algorithm with time-complexity $O(|I|^{f(t)})$ where f is a function and $|I|$ denotes the size of an instance I of the problem. A (maximization) problem is ρ-approximable, for some $\rho < 1$, if there exists an algorithm which guarantees that its solution value is always more than ρ times the value of an optimal solution.

In what follows, we consider a graph $G = (V, E)$ with $|V| = n$ and $|E| = m$. We denote by $\delta(G)$, $\Delta(G)$ and $\bar{\Delta}(G)$ (or simply δ, Δ and $\bar{\Delta}$) the minimum, maximum and average degree, respectively, of G. The diameter $\mathcal{D}(G)$ of a graph G is the length of the longest shortest-path over any pair of vertices of the graph. For a graph G, we denote by $tw(G)$ and $\chi(G)$ its treewidth and chromatic number, respectively. Given two sets of vertices $A, B \subseteq V$, $G[A]$ denotes the subgraph induced by A, $E(A)$ the set of edges in $G[A]$ and $E(A, B)$ the set of edges with one endpoint in A and the other one in B. Finally, throughout the paper we use the notation $O^*(\cdot)$ that ignores polynomial factors in the complexity expressions.

2 Related Work and Our Contribution

In [3] was proved that it is NP-complete even to decide if there is a DENSEST k-SUBGRAPH with at least $k^{1+\epsilon}$ edges, for any $\epsilon > 0$. Moreover, Cai in [15] proved that DENSEST k-SUBGRAPH is W[1]-hard, with respect to k even for regular graphs. This result immediately implies also that DENSEST k-SUBGRAPH is W[1]-hard with respect to the size ℓ of the solution, as any solution cannot contain more than $k(k-1)/2$ edges. On the other hand, DENSEST k-SUBGRAPH can be solved in time $O^*(kn^{\omega\lfloor k/3\rfloor+1+(k \mod 3)})$ where $\omega < 2.376$, by the exact algorithm proposed in [15]. Notice, however, that this algorithm requires exponential space. After the conference version of our paper, Chang et al. [17] presented an algorithm for finding a DENSEST k-SUBGRAPH which runs in $O^*(1.7315^n)$ time. Moreover, some results using the degeneracy of the input graph as a parameter have been proposed in [35].

The approximability of DENSEST k-SUBGRAPH has been also extensively studied. For instance, an approximation algorithm achieving ratio $8k/9n$ has been proposed in [6]. In [24], three procedures are used in order to obtain a $O(n^{-1/3})$-approximation ratio, while

the best known approximation algorithm achieves a ratio of $O(n^{-((1/4)+\epsilon)})$ within $n^{O(1/\epsilon)}$ time, for any $\epsilon > 0$ [5]. A polynomial time approximation scheme (PTAS) has been presented in [2] for a class of dense graphs known as *everywhere-dense* graphs, while several approximation results are known for special graph classes like bipartite graphs [4], chordal graphs [36] and interval graphs [39]. Moreover, the case where $k = n/2$ has been extensively studied in the literature (see for example [25, 31]). From a negative point of view, DENSEST k-SUBGRAPH in general graphs does not admit a PTAS, under the assumption that the problems in NP do not admit randomized algorithms that run in subexponential time [33], while it remains NP-hard even for bipartite graphs of maximum degree 3 [26]. DENSEST k-SUBGRAPH can be optimally solved in polynomial time in few graph classes, including k-trees and split graphs [19], as well as, graphs of maximum degree two [26]. The above results leave a large gap concerning the approximability of the problem both in general and bipartite graphs.

Several relaxed versions of the DENSEST k-SUBGRAPH problem have been studied in the literature. Specifically, if there is no constraint in the number of vertices of the solution sought, the DENSEST SUBGRAPH of a graph G is defined as a subgraph induced by a set of vertices A which maximizes the density of $G[A]$, that is the ratio $E(A)/|A|$. Note that, the density of a graph is equal to its average degree divided by two. A polynomial time algorithm based on maximum flow computations has been presented in [30] for finding the DENSEST SUBGRAPH of any input graph. Moreover, the DENSEST AT-MOST-k-SUBGRAPH (resp. DENSEST AT-LEAST-k-SUBGRAPH) of a graph G is the highest density subgraph of G induced by a set of vertices A so that $|A| \leqslant k$ (resp. $|A| \geqslant k$). Khuller and Saha [34] showed that a ρ-approximation algorithm for DENSEST AT-MOST-k-SUBGRAPH will imply a $\rho/4$-approximation algorithm for DENSEST k-SUBGRAPH, indicating that the former problem seems to be as difficult as the latter one. On the other hand, although finding a DENSEST AT-LEAST-k-SUBGRAPH is a NP-hard problem [34], several constant factor approximation algorithms have been proposed for it (see for example [1, 34]).

Furthermore, the DENSEST k-SUBGRAPH problem is a special case of the QUADRATIC 0–1 KNAPSACK problem which has been introduced in [29]. In the classical KNAPSACK problem each item is associated with a profit and a weight, and we search for the subset of items whose total weight does not exceed the KNAPSACK capacity b while their total profit is maximum. In the QUADRATIC 0–1 KNAPSACK problem the appearance of pairs of items in the KNAPSACK gives an additional profit. Specifically, for each pair of items we are given a profit which contributes to the objective if both items are included to the KNAPSACK. DENSEST k-SUBGRAPH is a special case of QUADRATIC 0–1 KNAPSACK if we associate each vertex to an item, and hence edges correspond to pairs of items. Moreover, the profit of each item is equal to zero, while the profit of each pair of items is equal to one if the corresponding edge exists, and zero otherwise. Finally, the weight of each item is equal to one and $b = k$. Taylor [43] presented a general transformation of any $O(n^{-(\rho+\epsilon)})$-approximation algorithm with running time $O(n^{1/\epsilon})$ for DENSEST k-SUBGRAPH to an $O(n^{-((2\rho)/(1+\rho))+3\epsilon})$-approximation algorithm with running time $O(n^{3/\epsilon})$ for QUADRATIC 0–1 KNAPSACK, for any $\rho \in (0, 1)$ and $\epsilon > 0$. Note that, very recently, an FPTAS for the QUADRATIC 0–1 KNAPSACK problem has been presented in [40] for graphs of bounded treewidth.

In this paper, we present (sub)exponential and parameterized algorithms that compute optimal or approximate solutions for the DENSEST k-SUBGRAPH problem.

More specifically, in Section 3 we propose exact algorithms for finding an optimal solution to DENSEST k-SUBGRAPH. The interest of these algorithms is that they link computation of an optimum for DENSEST k-SUBGRAPH to the computation of other graph-

parameters such as dominating set, vertex cover, independent dominating set, longest path, etc., exhibiting in this way interesting structural relations between k-densest subset and each of these parameters. These algorithms improve the trivial complexity $O^*(2^n)$ for the problem and, in contrast to the existing algorithms, they need only polynomial space. In this direction, we first present a general decomposition schema which, depending on the way the graph is decomposed, leads to different time complexities for finding an optimal solution. Let us note that this schema is quite general and can be applied to solve a lot of graph-problems, in particular problems whose feasible solutions are subsets of the vertex-set of the input graph (fixed-cardinality problems are such problems). The interesting algorithmic point of this technique is that it can avoid a complete enumeration of k-element vertex subsets, when it can be restricted to subsets $X \subseteq V$ such that $V \setminus X$ induces a graph of maximum degree at most 2. Next, in Section 3.2, we propose a branch-and-cut algorithm and we analyze its complexity using the "measure and conquer" and the "bottom-up" techniques. This kind of analysis gives more refined running time upper bounds and improves the existing ones in sparse graphs. Finally, in Section 3.3, we show that DENSEST k-SUBGRAPH is FPT with respect to the size τ of a minimum vertex cover of the input graph using only polynomial space.

In Section 4, we first present two XP-approximation schemata for DENSEST k-SUBGRAPH whose approximation ratio depends on their complexity. We next devise approximation algorithms that run in moderately exponential or parameterized time.

Finally, we conclude in Section 5.

3 Exact and parameterized algorithms

3.1 A decomposition technique

In this subsection we present a general technique for designing exact algorithms for DENSEST k-SUBGRAPH. The basic idea is to construct a bounded size subgraph, try all vertex subsets of it as partial solutions, and complete each partial solution optimally based on some properties of the remaining graph. This general technique can be also applied to other problems and especially to the family of fixed cardinality problems, like MAX k-COVER, SPARSEST k-SUBGRAPH, etc. However, for each of these problems, specific properties of the remaining graph should be defined in order to be able to complete the partial solutions in an optimal way.

More formally, given a graph $G = (V, E)$, we split the vertex set V into two subsets V_1 and V_2. Then, for each j, $0 \leqslant j \leqslant k$, and each subset $A_1 \subseteq V_1$ with $|A_1| = j$, we search for a subset $A_2 \subseteq V_2$, $|A_2| = k - j$, such that the number of edges in $G[A_1 \cup A_2]$ is maximized. Clearly, the complexity of this algorithm depends on:

- the size of the set V_1 as we need to try all subsets of V_1;
- the complexity of determining, given the set A_1, the appropriate set $A_2 \subseteq V_2$.

Hence, it is required for $V_2 = V \setminus V_1$ to have some specific property that allows A_2 to be determined in polynomial time. We will show that $\Delta(G[V_2]) \leqslant 2$ is such a property for DENSEST k-SUBGRAPH.

As we will see in what follows, this method provides a general framework for the complexity analysis of several algorithms (depending on the way V_1 is chosen and on its size), and uses polynomial space. Moreover, it is used later in Sections 3.3 and 4.2 for handling parametrization of DENSEST k-SUBGRAPH by the vertex cover number and for obtaining superpolynomial approximation results, respectively.

GENERIC(V_1, V_2) is a procedure that takes as input a partition of the vertex set (V_1, V_2) and returns an optimal DENSEST k-SUBGRAPH in G through exhaustive search.

Generic(V_1, V_2)

1: **for** $j = 0$ to k **do**
2: **for** any subset $A_1 \subseteq V_1$, such that $|A_1| = j$ **do**
3: find a solution $A = A_1 \cup A_2$ for the DENSEST k-SUBGRAPH problem in G such that $A_2 \subseteq V_2$, $|A_2| = k - j$, and $|E(A)|$ is maximized;
4: **return** the best among the solutions found in Line 3.

Whenever $\Delta(G[V_2]) \leqslant 2$, the following proposition states that A_2 can be found in polynomial time.

Proposition 1. *Consider a graph $G = (V, E)$, some partition of the vertex set V into two subsets V_1 and V_2 such that $\Delta(G[V_2]) \leqslant 2$, and a subset $A_1 \subseteq V_1$, $|A_1| \leqslant k$. Then a solution for the following problem Π: "determine a set $A_2 \subseteq V_2$ such that:*

- $|A_2| = k - |A_1|$
- $|E(A_1 \cup A_2)|$ *is maximized"*

can be found in $O(nk^2)$ time.

Proof. We will polynomially transform our problem to the QUADRATIC 0–1 KNAPSACK problem. Formally, the input of the QUADRATIC 0–1 KNAPSACK problem consists in an undirected graph $G = (V, E)$ and an integer b. A profit c_i and a weight w_i are associated with each vertex $i \in V$ and a profit c_{ij} is associated with each edge $(i, j) \in E$ where $i < j$. The goal is to find a subset $A \subseteq V$ such that the total weight of A does not exceed b, i.e., $\sum_{i \in A} w_i \leqslant b$, and the total profit of A, $\sum_{i \in A} c_i + \sum_{i,j \in A:\ i<j} c_{ij}$, is maximized. It has been shown in [41] that this problem can be solved in $O(|V|b^2)$ time for *edge series-parallel* (ESP) graphs. Each ESP graph G consists of a left and a right terminal vertex and is constructed by a series or a parallel operations performed on two ESP graphs G_1 and G_2. A series operation identifies the right terminal of G_1 with the left terminal of G_2, while a parallel operation identifies the left terminal of G_1 with the left terminal of G_2 and the right terminal of G_1 with the rights terminal of G_2. A single edge is the smallest ESP graph. For more details, see [41].

A graph of maximum degree 2 is a collection of cycles, paths and single vertices. We extend each of them with a left and a right fictive terminal vertex. All resulting components are ESP graphs. Then, the collection is converted to a single ESP graph by successive parallel operations. In this way, we convert the graph $G[V_2]$ to an ESP graph and we consider each vertex $i \in V_2$ assigned with a profit $c_i = |E(\{i\}, A_1)|$ and each edge $(i, j) \in E(V_2)$ assigned with a profit $c_{ij} = 1$. The fictive terminals and their incident edges are assigned with profits equal to zero. Furthermore, each vertex $i \in V_2$ is assigned with a weight $w_i = 1$, while fictive terminal vertices have an infinite weight. Finally, the capacity of the knapsack is $b = k - |A_1|$. An optimal solution to this QUADRATIC 0–1 KNAPSACK problem and hence to Π in G can be found within $O(nk^2)$ time [41]. □

Note that, if $\Delta(G[V_2]) = 0$, i.e., V_2 is an independent set, then the set A_2 can be found in $O(n \log k)$ time, by selecting the $k - |A_1|$ vertices of V_2 with the largest degree to A_1.

Proposition 2. GENERIC(V_1, V_2) *returns an optimal* DENSEST k-SUBGRAPH*-solution on $G[V_1 \cup V_2]$ in $O^*\left(\min\left\{\binom{|V_1|}{k}, 2^{|V_1|}\right\}\right)$ time whenever $A_2 \subseteq V_2$ can be computed in polynomial time.*

Proof. In some iteration, A_1 will be the subset of vertices of V_1 contained in an optimal solution. By Proposition 1, the whole solution $A_1 \cup A_2$ returned in such an iteration is an optimal one.

Line 3 of GENERIC(V_1, V_2) is executed $\sum_{j=0}^{k} \binom{|V_1|}{j} = O^*\left(\min\left\{\binom{|V_1|}{k}, 2^{|V_1|}\right\}\right)$ times. Hence, the complexity of the algorithm follows. □

Note that, if $k < |V_1|/2$ then $O^*\left(\binom{|V_1|}{k}\right)$ is strictly smaller than $O^*\left(2^{|V_1|}\right)$.

The following theorem handles four decompositions (V_1, V_2) of G, each one determined by the way V_1 is obtained. Other decompositions, based upon specific structural properties of the set V_1, can be also used to obtain different complexities.

Theorem 1. GENERIC(V_1, V_2) *leads to a polynomial space algorithm for* DENSEST k-SUBGRAPH *of time complexity:*

(i) $O^*\left(2^{\left(1-(5/8)^{\Delta-2}\right)\cdot n}\right)$, *if* V_1 *is obtained by repeated removals of sets dominating all vertices of degree at least 3;*

(ii) $O^*\left(2^{((\chi-1)/\chi)\cdot n}\right)$ *or* $O^*\left(2^{((\Delta-1)/\Delta)\cdot n}\right)$, *if* V_1 *is a minimum vertex cover;*

(iii) $O^*\left(2^{((\Delta-2)/(\Delta-1))\cdot n}\right)$, *for any* $\Delta \geqslant 3$, *if* V_1 *is obtained by repeated removals of minimum independent dominating sets;*

(iv) $O^*\left(2^{n-\mathcal{D}(G)}\right)$, *for any* $\Delta \geqslant 3$, *if* V_1 *is the complement of the vertices of a longest path of the graph.*

Proof of Item (i). We propose Algorithm 1 which works as follows: initially, we search for a set D_1 that dominates all vertices of degree at least 3 in $G = (V, E)$. By removing D_1 from V we get the induced subgraph $G[V \setminus D_1]$ of maximum degree at most $\Delta - 1$. Then, we search for a new set D_2 that dominates all vertices of degree at least 3 in $G[V \setminus D_1]$, we remove it from the graph, and we continue in the same way until the maximum degree of the remaining graph is at most 2. This is true after at most $\Delta - 2$ iterations of the above procedure. As the maximum degree of the final subgraph $G[V \setminus \bigcup D_i]$ is at most 2, we can apply GENERIC$(\bigcup D_i, V \setminus \bigcup D_i)$.

Algorithm 1 Decomposition by sets dominating vertices of degree at least 3

1: $V_1 = V$; $i = 1$;
2: **while** $\Delta(G[V_i]) > 2$ **do**
3: find a set $D_i \subseteq V_i$ that dominates all vertices of degree at least 3;
4: $V_{i+1} = V_i \setminus D_i$; $i = i + 1$;
5: **return** GENERIC$(\bigcup D_i, V \setminus \bigcup D_i)$.

The following claim shows the existence of the set D_i computed in Line 3 of Algorithm 1 and provides an upper bound to its size. Note that the proof of the claim is constructive.

Claim 1. *For any graph* $G = (V, E)$ *there exists a set* D *of size* $|D| \leqslant (3|V|/8) + O(1)$ *that dominates all vertices of degree at least 3.*

Proof of Claim 1. Our proof is based upon Reed's theorem [42] for the size of a minimum dominating set:

> *"any graph* $G = (V, E)$ *of minimum degree at least 3 has a dominating set of size at most* $3|V|/8$*".*

In order to apply Reed's theorem, we need to guarantee that the graph has minimum degree at least 3. Without loss of generality, we assume that G does not contain any connected component in which all vertices have degree at most 2. If this is not the case, we can remove these components from G as we do not care to dominate their vertices.

We extend the graph G by greedily adding fictive edges between the vertices of degree at most 2, until as many as possible of them have degree at least 3. At the end of this procedure either (a) there is one vertex of degree 1 or (b) there is one vertex of degree 2 or (c) there are two adjacent vertices both of degree 2 or (d) there are two adjacent vertices of degrees 1 and 2 or (e) all vertices have already degree at least 3. In cases (a)–(d), we continue adding fictive edges between vertices that still have degree smaller than 3 and vertices of degree at least 3 such that all vertices have degree at least 3 at the end. Note that, during this last step, we need to add at most three more fictive edges while at most two vertices of degree at least 3 in the original graph G are used. Let $G' = (V, E')$ be the resulting graph, which, by construction, has minimum degree at least 3. Thus, according to Reed's theorem, there is a dominating set D' in G' of size $|D'| \leqslant 3|V|/8$, i.e, D' dominates all vertices in G'. Hence, a dominating set D^* of minimum size in G' contains at most $3|V|/8$ vertices. However, at most two of the vertices of degree at least 3 in G may be dominated in D^* through fictive edges. For this reason, we extend D^* to D by including these vertices in the last one. Hence, D dominates all vertices of degree at least 3 in G while $|D| \leqslant |D^*| + 2 \leqslant (3|V|/8) + O(1)$, and the claim follows. □

Claim 2. *After the end of the while loop of Algorithm 1 it holds that:*

$$\left|\bigcup D_i\right| \leqslant \left(1 - \left(\frac{5}{8}\right)^{\Delta-2}\right)|V| + O(1)$$

Proof of Claim 2. By Claim 1, for each $i \geqslant 2$ we have:

$$|V_i| = |V_{i-1}| - |D_{i-1}| \geqslant |V_{i-1}| - \left(\frac{3\,|V_{i-1}|}{8} + O(1)\right) = \frac{5\,|V_{i-1}|}{8} - O(1)$$

As the while loop is executed at most $\Delta - 2$ times, by solving this recurrence for $i = \Delta - 2$ we get:

$$|V_{\Delta-2}| \geqslant \left(\frac{5}{8}\right)^{\Delta-2}|V| - \sum_{j=0}^{\Delta-3}\left(\frac{5}{8}\right)^{j} \cdot O(1) \geqslant \left(\frac{5}{8}\right)^{\Delta-2}|V| - O(1)$$

Hence, it holds that:

$$\left|\bigcup D_i\right| = |V| - |V_{\Delta-2}| \leqslant |V| - \left(\left(\frac{5}{8}\right)^{\Delta-2}|V| - O(1)\right)$$

and the proof of the claim is completed. □

To complete the proof for Item (i), note that the while loop is executed $\Delta - 2$ times and in each iteration a minimum dominating set is computed. As shown in Claim 1, the time-complexity of computing a set dominating all vertices of degree at least 3 is the same as the time-complexity of finding a minimum dominating set in a graph of the same vertex set. A minimum dominating set can be found in $O^*(1.5048^n)$ time [44], while, for graphs of maximum degree 3, this can be done in $O^*(1.202^n)$ time [28]. Note that, the space complexity of both algorithms is polynomial. According to Claim 2 the size of the set V_1 that takes as input $\textsc{Generic}(V_1, V_2)$ is $|V_1| \leqslant \left(1 - (5/8)^{\Delta-2}\right) n + O(1)$. Note

that, for $\Delta = 3$ it holds that $2^{1-(5/8)^{\Delta-2}} = 2^{3/8} > 1.202$, while for $\Delta \geqslant 4$ it holds that $2^{1-(5/8)^{\Delta-2}} \geqslant 2^{1-(5/8)^2} > 1.5048$. Moreover, by the definition of Algorithm 1 it holds that $G[V \setminus \bigcup D_i]$ is a graph of maximum degree 2. Therefore, using Propositions 1 and 2 the proof of Item (i) follows. □

Proof of Item (ii). In Algorithm 2 we use the fact that a minimum vertex cover B of size τ, can be found in time $O^*(1.2738^\tau)$ using polynomial space [18]. Then, the set $V \setminus B$ is a maximum independent set of size at least n/χ, since the vertex set of the input graph can be partitioned into χ independent sets. Hence, by Propositions 1 and 2 the first part of Item (ii) of the theorem follows.

Algorithm 2 Decomposition by minimum vertex cover

1: find a minimum vertex cover B of G;
2: **return** GENERIC$(B, V \setminus B)$.

If the input graph is a clique or an odd cycle, then the DENSEST k-SUBGRAPH problem is polynomial. Otherwise, by Brooks' Theorem [14], $\chi \leqslant \Delta$ and the second part of Item (ii) of the theorem follows. □

Proof of Item (iii). In Algorithm 3, we search for $\Delta - 2$ disjoint independent dominating sets $D_3, D_4, \ldots, D_\Delta$ of the input graph. If there exits a D_i such that $|D_i| \geqslant n/(\Delta-1)$, then we call GENERIC$(V \setminus D_i, D_i)$. Otherwise, we call GENERIC$(D, V \setminus D)$, where $D = \bigcup D_i$.

Algorithm 3 Decomposition by minimum independent dominating set

1: $V_\Delta = V$; $D = \emptyset$;
2: **for** $i = \Delta$ to 3 **do**
3: find an independent dominating set D_i on $G[V_i]$;
4: $D = D \cup D_i$; $V_{i-1} = V_i \setminus D_i$;
5: **if** there exists a D_i such that $|D_i| \geqslant n/(\Delta-1)$ **then**
6: **return** GENERIC$(V \setminus D_i, D_i)$;
7: **else**
8: **return** GENERIC$(D, V \setminus D)$.

An independent dominating set in Line 3 of Algorithm 3 can be found in $O^*(1.3351^n)$ time [8]. Note that this algorithm uses polynomial space.

In the case where there exists a D_i such that $|D_i| \geqslant n/(\Delta-1)$, we have that $|V \setminus D_i| \leqslant n - (n/(\Delta-1)) = ((\Delta-2)/(\Delta-1)) \cdot n$. Moreover, as D_i is an independent set, we can apply GENERIC and get the stated complexity.

Otherwise, for each i, $3 \leqslant i \leqslant \Delta$, it holds that $|D_i| \leqslant n/(\Delta-1)$, and hence, $|D| \leqslant (\Delta - 2) \cdot (n/(\Delta-1))$. Since $G[V \setminus D]$ is a graph of maximum degree 2, we can apply Proposition 2, completing the proof of Item (iii). □

Proof of Item (iv). To prove Item (iv) of the theorem, we propose another algorithm that is obtained by considering the diameter $\mathcal{D}(G)$ of the input graph.

Note first, that P contains the vertices of a maximum shortest path of the graph, that is the vertices that define the diameter of the graph. Since T contains edges only between two consecutive levels, $G[P]$ is a path, that is a graph of maximum degree 2. As $|P| \geqslant \mathcal{D}(G)$, it holds that $|V \setminus P| \leqslant n - \mathcal{D}(G)$. Therefore, by applying Proposition 2 the proof of Item (iv) and of the theorem are completed. □

Algorithm 4 Decomposition by the diameter

1: **for** each $v \in V$ **do**
2: build a BFS tree T rooted at v;
3: determine a longest path, P_v, from v in T;
4: $P = \{P_v : |P_v| \text{ is maximum}, v \in V\}$;
5: **return** GENERIC$(V \setminus P, P)$.

Let us now consider the DENSEST k-SUBGRAPH problem in bipartite graphs. Consider a bipartite graph $G = (U \cup V, E)$ and assume, without loss of generality, that $|U| \leqslant n/2 \leqslant |V|$. Fix an optimal solution $S^* = A_1^* \cup A_2^*$ for DENSEST k-SUBGRAPH in G where $A_1^* \subseteq U$ and $A_2^* \subseteq V$. Revisit GENERIC(U, V) and observe that once A_1^* is given, the completion of S^*, i.e., the computation of A_2^* in Line 3 can be performed in polynomial time. Indeed, given A_1^*, one can consider the vertices of V in decreasing order with respect to the cardinality of the intersection of their neighborhood with A_1^* and take the $k - |A_1^*|$ first vertices of V. So, the running time of GENERIC(U, V) is $\sum_{i \leqslant k} \binom{|U|}{i}$. Then, GENERIC$(U, V)$ takes time:

$$\sum_{i \leqslant k} \binom{\frac{n}{2}}{i} \leqslant O^* \left(\binom{\frac{n}{2}}{k} \right) \leqslant O^* \left(2^{n/2} \right)$$

The above complexity can be further improved by carefully taking into account the balance between subsets of U and V in an optimal solution. In order to do this, let $\phi(b, |X|)$ be the worst-case complexity of GENERIC(X, Y) if we allow the optimal solution to contain at most b vertices from X. Assume again that $|U| \leqslant n/2 \leqslant |V|$ and let $\lambda = |U|/n \leqslant 1/2$. In general, GENERIC$(U, V)$ finds a DENSEST k-SUBGRAPH in $O^*(\phi(k, \lambda n))$ time, while GENERIC(V, U) finds it in time $O^*(\phi(k, (1-\lambda)n))$. Fix now some scalar $\nu \leqslant 1/2$. Then, either V contains at most νk vertices from an optimal solution and U more than $(1-\nu)k$ of them, or V contains more than νk vertices from an optimal solution and U contains at most $(1-\nu)k$ of them. Hence, we only need to consider small subsets from U and V. The overall running time is then:

$$T(n) \leqslant \max_{\lambda} \min_{\nu} \{\phi(\nu k, (1-\lambda)n) + \phi((1-\nu)k, \lambda n)\}$$

Since the min argument in $T(n)$ involves an increasing and a decreasing function in k and n, it is easy to find the solution of this minimization problem for a given set of parameters (k, n). However, it would be very tedious to try giving an exact formula considering all the specific cases when k is close to n. Instead, in the following corollary we give a sample of $T(n)$ values.

Corollary 1. *There is an algorithm that finds a* DENSEST k-SUBGRAPH *in a bipartite graph in time $T(n)$, where $T(n)$ is given in the following table for different values of k/n:*

k/n	$1/100$	$1/20$	$1/10$	$1/6$	$1/4$	$1/3$
$T(n)$	1.029^n	1.105^n	1.177^n	1.253^n	1.325^n	1.375^n
$\sum_{i \leqslant k} \binom{n/2}{i}$	1.051^n	1.177^n	1.285^n	1.375^n	$\sqrt{2}^n$	$\sqrt{2}^n$

3.2 Branch-and-cut algorithms

In this section we propose two slightly different branching algorithms for DENSEST k-SUBGRAPH and we prove upper bounds on their time complexities. For the analysis of the first algorithm we use the well known technique of *measure and conquer* introduced

in [27]. For the analysis of the second algorithm we use the *bottom-up* technique which has been developed in [10] as a technique for more carefully measuring the progression of a branching algorithm. This method has been used in [10] for the independent set problem deriving an upper complexity bound that has remained for long time the best known for this problem. It has been also used in [11] for the quasi-independent set problem.

Let us first consider a *simple branch-and-cut algorithm* that branches on a vertex of maximum degree. The branching tree is pruned whenever the remaining graph is of maximum degree 2. In this case, a solution for the whole graph can be obtained by extending the solution implied by the particular path of the search tree as stated in Proposition 1.

Theorem 2. *The simple branch-and-cut algorithm finds a* DENSEST k-SUBGRAPH *in* $O^*\left(2^{((\Delta-1)/(\Delta+1))\cdot n}\right)$ *time.*

Proof. In order to refine the trivial analysis of the simple branch-and-cut algorithm that leads to time complexity of $O^*(2^n)$, our analysis uses the measure and conquer technique [27]. Following this technique, we will describe how to assign a weight $w(v)$ for each vertex v in the remaining graph, i.e., the graph induced by the vertices which we have not yet been used to branch on them. Let V_i, $0 \leqslant i \leqslant \Delta$, be the set of vertices of degree i in the remaining graph and w_i be a weight associated to each degree i which will be defined later. Then, each vertex $v \in V_i$ is assigned a weight $w(v)$ according to its degree in the remaining graph; more specifically we set $w(v) = w_i$. The weights w_i will be selected once in such a way that satisfy the following constraints:

- w_i is increasing with i;
- $w_{i+1} - w_i$ is decreasing with i; this convexity hypothesis is necessary for accurate assessment of worst-case branching.

The analysis for the complexity of the simple branch-and-cut algorithm will be based on the quantity $W = \sum_{v \in V} w(v) = \sum_{i=0}^{\Delta} |V_i| w_i$.

Consider an iteration and assume that the algorithm branches on a vertex v of maximum degree i in the remaining graph. For each neighbor u of v, its degree $d(u)$ decreases by 1 in the remaining graph after the branching on v with respect to its degree before branching on v, since it looses one neighbor. Hence, due to the branching on v the total weight W decreases. Specifically, we have:

$$T(W) \leqslant 2T\left(W - w_i - \sum_{u \in N(v)} \left(w_{d(u)} - w_{d(u)-1}\right)\right) \leqslant 2T\left(W - w_i - i\left(w_i - w_{i-1}\right)\right)$$

where the last inequality follows by the convexity hypothesis.

In fact, we only need to verify the above expression for $i \geqslant 3$, since, by Proposition 1, we can complete the solution in polynomial time if the remaining graph has maximum degree 2. However, we are not allowed to set $w_i = 0$ for $i \leqslant 2$, since we need to verify also the convexity hypothesis. It is easy to see that the following choices are optimal:

- $w_0 = 0$: disconnected vertices have no influence on the branching;
- $w_1 = {}^{w_3}/_3$ and $w_2 = 2 \cdot {}^{w_3}/_3$: indeed, the exact value of w_1 has no influence on the complexity, while the smaller w_2 the better the complexity;
- for each i, $3 \leqslant i \leqslant \Delta$, it should hold that $(i+1)w_i - iw_{i-1} = c$, for some c that we have to define.

For each i, $3 \leqslant i \leqslant \Delta$, we sum up the latter equations for $j = 3, \ldots, i$ and we get $w_i = {}^{((i-2)c+3w_2)}/_{(i+1)}$. Then, we have $w_3 = {}^{c}/_{2}$, $w_2 = {}^{c}/_{3}$ and $w_1 = {}^{c}/_{6}$. Furthermore, we are free to set $w_\Delta = 1$ (which gives also $W \leqslant n$) and thus we get $c = {}^{(\Delta+1)}/_{(\Delta-1)}$. Hence, for each $i \geqslant 2$ we have $w_i = {}^{((i-1)(\Delta+1))}/_{((i+1)(\Delta-1))}$.

Using the above definition of weights, the recurrence inequality admits as a solution $T(n) \leqslant 2^{((\Delta-1)/(\Delta+1))\cdot n}$ and the theorem follows. □

We now slightly modify the previous basic branching algorithm by using a different method as soon as the remaining graph has average degree 3. The analysis of this modified branching algorithm is based on the bottom-up method. The following Lemma 1 settles the case where the average degree of the graph is at most 3, while Lemma 2 handles the complexity of finding a DENSEST k-SUBGRAPH on graphs with average degree at least $d-1$, given that the complexity of finding a DENSEST k-SUBGRAPH for graphs with average degree at most $d-1$ is known.

Lemma 1. DENSEST k-SUBGRAPH *can be solved on graphs of average degree* $\bar{\Delta} \leqslant 3$ *with running time* $O^*(2^{21n/46})$.

Proof. If $\Delta \leqslant 3$, then by Theorem 1 Item (i), DENSEST k-SUBGRAPH can be solved in $O^*(2^{3n/8})$ time.

Otherwise, we make a sequence of branchings, each time choosing a vertex of maximum degree in the remaining graph, until we get a graph of maximum degree 3. Let n_i, $4 \leqslant i \leqslant \Delta$, be the number of vertices of degree i on which we have branched during the above procedure. In other words, n_i corresponds to the number of iterations in which we branch on a vertex of degree i. Let $n' = \sum_{i=4}^{\Delta} n_i$. Since i is the degree at the time we branch, the number of deleted edges is $\sum_{i=4}^{\Delta} i \cdot n_i \geqslant 4n'$. For the $n - n'$ remaining vertices of the graph we proceed as follows:

(α) If $n - n' < {}^{20n}/_{23}$, greedily branch on vertices of degree 3 until the graph has maximum degree 2. Then, the running time of our algorithm is $O^*(2^x)$ where:

$$x \leqslant n' + \frac{m-4n'}{3} = \frac{m}{3} - \frac{n'}{3} \leqslant \frac{n}{2} - \frac{1}{3}\cdot\frac{3n}{23} = \frac{21n}{46}$$

(β) If $n - n' \geqslant {}^{20n}/_{23}$, compute a minimum dominating set as described in Algorithm 1 and branch on any vertex of it. Then, the running time is $O^*(2^x)$ where:

$$x \leqslant n' + \frac{3(n-n')}{8} = \frac{3n}{8} + \frac{5n'}{8} \leqslant \frac{3n}{8} + \frac{5}{8}\cdot\frac{3n}{23} = \frac{21n}{46}$$

The proof of the lemma is now complete. □

Lemma 2. *Assume* DENSEST k-SUBGRAPH *can be optimally solved in graphs with average degree at most* $d-1$, *in time* $O^*(2^{\alpha_d n})$ *for some* $\alpha_d \geqslant {}^{1}/_{2}$, $d \in \mathbb{N}$. *Then its computation time:*

- *in graphs with average degree at least* $d-1$ *is* $O^*(2^{\alpha_d n + \beta_d(m-(d-1)\cdot n/2)})$, *where* m *is the number of edges,* $\beta_d = {}^{2(1-\alpha_d)}/_{(d+1)}$;
- *in graphs with average degree at most* d, *this time is* $O^*(2^{\alpha_{d+1} n})$, *where* α_{d+1} *is defined as:* $\alpha_{d+1} = {}^{(d\alpha_d+1)}/_{(d+1)}$.

Proof. We proceed by induction on both n and m: we assume that it is true for any $n' < n, m' \in \mathbb{N}$ and for $n' = n, m' < m$ and we deduce that it remains true for n, m. For every value of n, the claim is true for $m \leqslant (d-1) \cdot n/2$, by the hypothesis of our lemma. So, we consider values of m greater than $(d-1) \cdot n/2$. This means that there exists some vertex of degree at least d. When branching on such a vertex, we get:

$$\begin{aligned} T(m,n) &\leqslant 2T(m-d, n-1) \leqslant 2^{1+\alpha_d(n-1)+\beta_d\left(m-\frac{(d-1)n}{2}\right)-\frac{d+1}{2}} \\ &\leqslant 2^{1-\alpha_d-\frac{d+1}{2}\cdot\frac{2(1-\alpha_d)}{d+1}} \cdot 2^{\alpha_d n+\beta_d\left(m-\frac{(d-1)n}{2}\right)} \leqslant 2^{\alpha_d n+\beta_d\left(m-\frac{(d-1)n}{2}\right)} \end{aligned}$$

This, by induction, remains true for any m, n. In particular, if the average degree is less than, or equal to, d, then $m \leqslant dn/2$; hence:

$$T(m,n) \leqslant 2^{\alpha_d n+\beta_d\left(\frac{dn}{2}-\frac{(d-1)n}{2}\right)} \leqslant 2^{\alpha_d n+\frac{2(1-\alpha_d)}{d+1}\cdot\frac{n}{2}} \leqslant 2^{n\left(\alpha_d+\frac{1-\alpha_d}{d+1}\right)} = 2^{n\cdot\frac{d\alpha_d+1}{d+1}} = 2^{\alpha_{d+1}n}$$

The proof of the lemma is now completed. □

Putting Lemmata 1 and 2 together, the following theorem can be derived.

Theorem 3. DENSEST k-SUBGRAPH *can be solved on graphs of average degree* $\bar{\Delta} \leqslant d$ *with running time* $O^*(2^{((d-27/23)/(d+1))\cdot n})$*, for any* $d \in \mathbb{N}, d \geqslant 3$.

Proof. For $d = 3$ the result follows from Lemma 1. Assume it is true for $\bar{\Delta} \leqslant d-1$. Then, from Lemma 2, we can find a solution when $\bar{\Delta} \leqslant d$ with running time $O^*(2^{\alpha_{d+1}n})$, where:

$$\alpha_{d+1} = \frac{d\alpha_d + 1}{d+1} = \frac{d\frac{d-1-\frac{27}{23}}{d} + 1}{d+1} = \frac{d-\frac{27}{23}}{d+1}$$

Thus, the statement holds by induction on d. □

3.3 Parameterization by minimum vertex cover size

Let us note that an easy application of the method of [38] for MAX k-COVER, derives that DENSEST k-SUBGRAPH can be solved in $O^*(2^{tw})$, using space exponential in tw [12]; in other words, DENSEST k-SUBGRAPH is FPT with respect to the treewidth of the input graph.

For the size τ of the minimum vertex cover of the input graph it holds that $tw \leqslant \tau$. So, the above remark implies that DENSEST k-SUBGRAPH is FPT with respect to τ too. In what follows, we present another application of GENERIC and we restate Item (ii) of Theorem 1 in order to obtain the following parameterized result with respect to τ which uses only polynomial space. In preamble, observe that a minimum vertex cover VC can be computed in polynomial space by the algorithm of [18], while GENERIC$(VC, V \setminus VC)$ just needs to keep in memory the best solution currently known.

Theorem 4. DENSEST k-SUBGRAPH *can be solved in* $O^*(2^\tau)$ *time using polynomial space.*

Let us remark that, since in a bipartite graph $\tau \leqslant n/2$, Theorem 4 has as immediate corollary that DENSEST k-SUBGRAPH *in bipartite graphs can be solved in time* $O^*(2^{n/2}) \approx O^*(1.414^n)$.

We now refine the analysis of Theorem 4 and prove that, informally, the instances of DENSEST k-SUBGRAPH that are not fixed-parameter tractable with respect to k are those solved with running time better than $O^*(2^\tau)$.

Theorem 5. *Given a constant* $\lambda \in (0, 1/2)$, DENSEST k-SUBGRAPH *can be solved in* $O^*(\max\{\gamma(\lambda)^\tau, c(\lambda)^k\})$ *time using polynomial space, where:*

$$\gamma(\lambda) = \frac{1}{\lambda^\lambda (1-\lambda)^{1-\lambda}} \quad \textit{and} \quad c(\lambda) = \frac{\left(\frac{1}{\lambda}\right)^{1/\lambda}}{\left(\frac{1}{\lambda}-1\right)^{\left(\frac{1}{\lambda}-1\right)}}$$

Proof. Since we use the same algorithm as in Theorem 4 the space complexity follows.

If $\tau \leqslant k$, then by Theorem 4 it follows that DENSEST k-SUBGRAPH can be solved in $O^*(2^\tau) = O^*(2^k)$ time. Hence, we can assume that $k < \tau$. We distinguish the following two cases: $\tau > k \geqslant \lambda\tau$ and $k < \lambda\tau$. Recall by the proof of Item (ii) in Theorem 1 which implies the proof of Theorem 4, that the more refined expression of the running time of the algorithm is $O^*\left(\sum_{i=1}^{k} \binom{\tau}{i}\right)$.

If $k \geqslant \lambda\tau$, then using the fact that $k < {}^{(k/\lambda)}/_2$ and Stirling's formula we get

$$\sum_{i=1}^{k} \binom{\tau}{i} \leqslant k\binom{\frac{k}{\lambda}}{k} \simeq k\left(\frac{(\lambda^{-1})^{\lambda^{-1}}}{(\lambda^{-1}-1)^{(\lambda^{-1}-1)}}\right)^k = O^*\left(c(\lambda)^k\right)$$

If $k < \lambda\tau$, then $k < {}^{\tau}/_2$ and hence:

$$\sum_{i=1}^{k} \binom{\tau}{i} \leqslant k\binom{\tau}{k} \leqslant k\left(\frac{1}{\lambda^\lambda (1-\lambda)^{(1-\lambda)}}\right)^\tau = O^*\left(\gamma(\lambda)^\tau\right)$$

and the theorem follows. □

The table below contains the values of $c(\lambda)$ and $\gamma(\lambda)$ for some values of λ.

λ	0.01	0.05	0.1	0.15	0.2	0.25	0.3	0.35	0.40	0.45	0.49
$c(\lambda) = \frac{\frac{1}{\lambda^{\frac{1}{\lambda}}}}{\left(\frac{1}{\lambda}-1\right)^{\left(\frac{1}{\lambda}-1\right)}}$	270.47	53.00	25.81	16.74	12.21	9.48	7.66	6.36	5.38	4.61	4.11
$\gamma(\lambda) = \frac{1}{\lambda^\lambda (1-\lambda)^{1-\lambda}}$	1.06	1.22	1.38	1.53	1.65	1.75	1.84	1.91	1.96	1.99	1.9996

4 Approximation algorithms

Up to now no constant factor approximation algorithm for DENSEST k-SUBGRAPH that runs in polynomial time is known. In this section, we relax polynomiality requirement and devise approximation algorithms that, although exponential, are provably faster than the exact algorithms for the problem.

This approach has already been considered for several other paradigmatic problems such as MINIMUM SET COVER [20], MIN COLORING [7], MAX INDEPENDENT SET and MIN VERTEX COVER [9], MIN BANDWIDTH [21], etc. Note that, the $O(n^{-((1/4)+\epsilon)})$-approximation algorithm with complexity $n^{O(1/\epsilon)}$ presented in [5], can be considered as an approximation algorithm in this context, since whenever ϵ is chosen to be of the form $\log_n c$, where c is a constant, a constant factor approximation ratio is achieved in subexponential time. Note finally that similar issues arise in the field of FPT algorithms, where approximation notions have been introduced, for instance, in [13, 16, 23, 37].

For better readability, we partition the results of this section into two parts. In the first part, we give approximation algorithms with complexity of the form $O^*(n^{ck})$, with $0 < c \leqslant 1$. In the second part, we present approximation algorithms that either have complexity of the form $O^*(c^n)$, with $1 < c \leqslant 2$, or they are FPT algorithms.

4.1 XP-approximation algorithms

A general idea for the design of an exponential time approximation algorithm is to construct a "good" subgraph of ρk vertices in exponential time and select the remaining $(1-\rho)k$ vertices in a greedy way. In this vein, the following proposition gives a property of such a good subgraph.

Proposition 3. *For an optimal solution A^* for* DENSEST k-SUBGRAPH *and a rational ρ such that $0 < \rho \leqslant 1$, there exists a partition of the vertices of A^* into two subsets A_1^*, $|A_1^*| = \rho k$, and A_2^*, $|A_2^*| = (1-\rho)k$, such that $|E(A_1^*)| \geqslant {}^{(\rho)}/_{(1-\rho)} \cdot |E(A_2^*)|$.*

Proof. Consider an arbitrary partition of the vertices in A^* into $\lceil {}^1/_{(1-\rho)} \rceil$ subsets B_i, $i = 1, \ldots, \lceil {}^1/_{(1-\rho)} \rceil$ each one of at most $\lceil (1-\rho)k \rceil$ vertices. Assume, w.l.o.g., $|E(B_1)| \geqslant |E(B_2)| \geqslant \ldots \geqslant |E(B_{\lceil 1/(1-\rho) \rceil})|$. Then, by considering $A_1^* = \bigcup_{i=1}^{\lceil 1/(1-\rho) \rceil - 1} B_i$ and $A_2^* = B_{\lceil 1/(1-\rho) \rceil}$, we have:

$$|E\left(A_1^*\right)| = \sum_{i=1}^{\lceil 1/(1-\rho) \rceil - 1} |E\left(B_i\right)| \geqslant \left(\left\lceil \frac{1}{1-\rho} \right\rceil - 1 \right) \cdot \left| E\left(B_{\lceil 1/(1-\rho) \rceil}\right) \right| \geqslant \frac{\rho}{1-\rho} \cdot |E\left(A_2^*\right)|$$

that completes the proof. □

Algorithm 5 Create all subsets

1: **for** each of the $\binom{n}{\rho k}$ subsets of vertices $A_1 \subseteq V$, $|A_1| = \rho k$, **do**
2: build $A_2 \in V \setminus A_1$, $|A_2| = (1-\rho)k$, having the highest degree to A_1;
3: **return** the maximum among the $A_1 \cup A_2$ computed.

Theorem 6. *For any ρ, $0 < \rho \leqslant 1$, Algorithm 5 achieves a ρ-approximation ratio in $O^*(n^{\rho k})$ time.*

Proof. In some iteration, the algorithm will consider as A_1 the set A_1^* of Proposition 3. In this iteration a solution of $|E(A_1)|+|E(A_2)|+|E(A_1,A_2)| = |E(A_1^*)|+|E(A_2)|+|E(A_1,A_2)|$ edges is built. Since A_2 contains the vertices of the highest degree to A_1^*, it holds that $|E(A_1,A_2)| \geqslant |E(A_1^*,A_2^*)|$. Therefore, using Proposition 3 we get approximation ratio:

$$\frac{|E\left(A_1^*\right)| + |E\left(A_2\right)| + |E\left(A_1^*, A_2^*\right)|}{|E\left(A_1^*\right)| + |E\left(A_2^*\right)| + |E\left(A_1^*, A_2^*\right)|} \geqslant \frac{|E\left(A_1^*\right)| + |E\left(A_1^*, A_2^*\right)|}{|E\left(A_1^*\right)| + \frac{1-\rho}{\rho} |E\left(A_1^*\right)| + |E\left(A_1^*, A_2^*\right)|} \geqslant \rho$$

The complexity of the algorithm is determined by the loop in Line 1 that iterates $\binom{n}{\rho k} = O(n^{\rho k})$ times. □

Another way to construct a good subgraph of ρk vertices is to run an exact algorithm for DENSEST ρk-SUBGRAPH and to complete the solution with $(1-\rho)k$ arbitrarily selected vertices. In the following lemma we use another definition of the density of a graph $G = (V, E)$. Specifically, we define $\varpi(G)$ (or simply ϖ) to be the ratio of the number of edges $|E|$ of G over the number of edges of a complete graph of $|V|$ vertices, i.e., $\varpi = {}^{2|E|}/_{|V|(|V|-1)}$. Then, the following lemma will be used to count the number of edges induced by an optimal DENSEST ρk-SUBGRAPH.

Lemma 3. *Consider a graph $G = (V, E)$ with $\varpi(G) = {}^{2|E|}/_{(|V|(|V|-1))}$. For any p, $2 \leqslant p \leqslant |V|$, there exists a set of vertices $V_p \subseteq V$, $|V_p| = p$, such that for the induced subgraph $G[V_p] = G_p(V_p, E(V_p))$ it holds that $\varpi(G[V_p]) = {}^{2|E(V_p)|}/_{(|V_p|(|V_p|-1))} \geqslant \varpi(G)$.*

Proof. Assume for contradiction that the statement of the lemma does not hold for several p's and let q be the maximum such value. Note that $q \leqslant n-1$, since for $p = n$ the statement holds by definition. Then, for any $v \in V_{q+1}$:

$$\begin{aligned} & \varpi(G[V_{q+1} \setminus \{v\}]) = \frac{2|E(V_{q+1} \setminus \{v\})|}{|V_{q+1} \setminus \{v\}|(|V_{q+1} \setminus \{v\}| - 1)} < \varpi(G) \\ \Leftrightarrow \quad & \varpi(G) \cdot \frac{q(q-1)}{2} > |E(V_{q+1} \setminus \{v\})| \end{aligned}$$

Summing up for all vertices in V_{q+1}, we get:

$$\begin{aligned} \frac{\varpi(G) \cdot q(q-1)(q+1)}{2} &> \sum_{v \in V_{q+1}} |E\left(V_{q+1} \setminus \{v\}\right)| \\ &= (q+1)\,|E\left(V_{q+1}\right)| - \sum_{v \in V_{q+1}} |\{(v,u) \in E, u \in V_{q+1}\}| \\ &= (q+1)\,|E\left(V_{q+1}\right)| - 2\,|E\left(V_{q+1}\right)| \\ &\geqslant (q-1)\frac{\varpi(G) \cdot q(q+1)}{2} \end{aligned}$$

a contradiction. □

In the following theorem, we assume that an algorithm of complexity $\phi(k,t)$ is known for finding a DENSEST k-SUBGRAPH, where t is some parameter of the instance, e.g., $t = \Delta, \tau, \ell, n$. This algorithm is used in order to obtain an optimal solution of size ρk for the problem, where $0 < \rho \leqslant 1$.

Theorem 7. *Let $\mathcal{A}$ be an exact algorithm of complexity $\phi(k,t)$ for finding a* DENSEST k-SUBGRAPH*, where t is a parameter of the instance. For any ρ such that $0 < \rho \leqslant 1$, it is possible to find a ρ^2-approximation for* DENSEST k-SUBGRAPH *in G with running time $O^*(\phi(\rho k, t))$.*

Proof. We use the algorithm $\mathcal{A}$ to find a DENSEST $(\lceil \rho k \rceil + 1)$-SUBGRAPH. Let $V' \subseteq V$, $|V'| = \lceil \rho k \rceil + 1$, be the solution obtained by $\mathcal{A}$ where $\varpi(G[V']) = {}^{2|E(V')|}\!/_{(|V'|(|V'|-1))}$.

Consider an optimal solution $A^* \subseteq V$, $|A^*| = k$, for the DENSEST k-SUBGRAPH problem where $\varpi(G[A^*]) = {}^{2|E(A^*)|}\!/_{(k(k-1))}$. Let $V'' \subseteq A^*$ be a subset of A^* such that $|V''| = \lceil \rho k \rceil + 1$ and $|E(V'')|$ is maximized. Let $\varpi(G[V'']) = {}^{2|E(V'')|}\!/_{(|V''|(|V''|-1))}$. Since $G[V']$ is a DENSEST $(\lceil \rho k \rceil + 1)$-SUBGRAPH and $|V'| = |V''| = \lceil \rho k \rceil + 1$, it holds that $\varpi(G[V']) \geqslant \varpi(G[V''])$. Moreover, by Lemma 3 we have $\varpi(G[V'']) \geqslant \varpi(G[A^*])$ since $V'' \subseteq A^*$. Hence, we have that $\varpi(G[V']) \geqslant \varpi(G[A^*])$ and we get:

$$|E(V')| \geqslant \frac{\rho k(\rho k - 1)}{k(k-1)}|E(A^*)| = \left(\rho^2 \cdot \frac{k}{k-1} - \frac{\rho}{k-1}\right)|E(A^*)|$$

Since $\rho \leqslant 1$ and $k-1$ can be considered arbitrarily large (if k is a fixed constant, DENSEST k-SUBGRAPH can be solved in polynomial time), the above expression is arbitrarily close to ρ^2.

Finally, by completing the solution V' with $k - \lceil \rho k \rceil - 1$ arbitrary vertices, the theorem follows. □

In Theorem 7, we count only the edges induced by a densest $(\lceil \rho k \rceil + 1)$-subgraph, as the remaining vertices are selected arbitrarily. In Algorithm 6, we replace this greedy step by searching for successive DENSEST $(\lceil \rho k \rceil + 1)$-SUBGRAPHS.

Algorithm 6 Approximate subsets

```
1: A = ∅; i = 1; G_i = G;
2: while |A| < k do
3:    compute a DENSEST (⌈ρk⌉ + 1)-SUBGRAPH in G_i;
4:    let A_i be the set of vertices of this subgraph;
5:    if |A ∪ A_i| ⩽ k then
6:       create the graph G_{i+1} by removing from G_i the edges of E(A_i);
7:       A = A ∪ A_i;
8:       if the vertices of G_{i+1} induce an independent set then
9:          arbitrarily complete A with vertices in V \ A such that |A| = k;
10:   else
11:      arbitrarily complete A with vertices in V \ A such that |A| = k;
12:      return A;
13:   i = i + 1;
14: return A.
```

Theorem 8. *Let $\mathcal{A}$ be an exact algorithm of complexity $\phi(k,t)$ for finding a* DENSEST k-SUBGRAPH. *For any ρ such that $0 < \rho \leqslant 2/3$, Algorithm 6 achieves a $\rho(1-(3\rho/2))$-approximation ratio for* DENSEST k-SUBGRAPH *in G with running time $O^*(\phi(\rho k,t))$.*

Proof. Let λ be the number of iterations of Algorithm 6. As at the beginning of each iteration there exists at least one edge in G_i, there exists also a vertex $v \in A_i$ such that $v \notin A$. Moreover, in each iteration, at most ρk new vertices are added in the solution. Thus, it holds that $1/\rho \leqslant \lambda \leqslant k(1-\rho)$. Therefore, the running time of the algorithm is bounded by $O^*(\phi(\rho k,t))$.

At the beginning of iteration $i+1$, $i \geqslant 1$, the current graph G_{i+1} contains at least $|E(A^*)| - |E_i|$ edges, where $|E_i| = \sum_{j=1}^{i} |E(A_j)|$ and A^* is an optimal solution for the DENSEST k-SUBGRAPH problem. Thus, by Theorem 7, there exists a subgraph of G_{i+1} with size ρk that contains at least $\rho^2(|E(A^*)| - |E_i|)$ edges. We prove by induction on i that:

$$|E_i| \geqslant \rho^2 \left(i - \left(\frac{i(i-1)}{2} \right) \rho^2 \right) |E(A^*)|$$

For $i = 1$, by Theorem 7 the inequality holds. Assume that it is true for $i-1$. Then:

$$\begin{aligned}
|E_i| - |E_{i-1}| &\geqslant \rho^2 \left(|E(A^*)| - |E_{i-1}| \right) \implies \\
|E_i| &\geqslant \rho^2 |E(A^*)| + (1-\rho^2)|E_{i-1}| \\
&\geqslant \rho^2 \left(1 + \left(1-\rho^2\right) \left(i - 1 - \frac{(i-1)(i-2)}{2} \rho^2 \right) \right) |E(A^*)| \\
&= \rho^2 \left(1 + i - 1 - \frac{(i-1)(i-2)}{2} \rho^2 - (i-1)\rho^2 + \frac{(i-1)(i-2)}{2} \rho^4 \right) |E(A^*)| \\
&= \rho^2 \left(i - (i-1) \left(\frac{(i-2)}{2} + 1 \right) \rho^2 + \frac{(i-1)(i-2)}{2} \rho^4 \right) |E(A^*)| \\
&\geqslant \rho^2 \left(i - \frac{i(i-1)}{2} \rho^2 \right) |E(A^*)|
\end{aligned}$$

Let $E(A)$ be the set of edges of the final solution obtained by the algorithm. As Algorithm 6

iterates at least $((1/\rho) - 1)$ times, we have:

$$|E(A)| \geqslant \rho^2 \left(\left(\frac{1}{\rho} - 1 \right) - \frac{\left(\frac{1}{\rho} - 1 \right) \left(\frac{1}{\rho} - 2 \right)}{2} \rho^2 \right) |E(A^*)| \geqslant \rho \left(1 - \frac{3\rho}{2} \right) |E(A^*)|$$

that completes the proof. □

In general, Algorithm 6 outperforms Algorithm 5 for small values of ρ ($\rho \leqslant 2/5$), since in that case $\rho(1 - 3 \cdot \rho/2)$ is close to ρ and $\mathcal{A}$ runs faster than exhaustive enumeration. Algorithm 5 outperforms Algorithm 6 whenever ρ is close to 1.

4.2 Parameterized and moderately exponential approximation

As already mentioned, DENSEST k-SUBGRAPH is not fixed parameter tractable with respect to k [15], and hence, neither with respect to the size of an optimal solution ℓ. However, in this section we show that there exists an approximation algorithm for DENSEST k-SUBGRAPH achieving non-trivial approximation ratios (though non-constants) unattainable in polynomial time, with complexity parameterized by k (and hence by ℓ).

Theorem 9. DENSEST k-SUBGRAPH *is approximable within any ratio $R(n)$, where R is any strictly increasing and reversible function, in parameterized time w.r.t. k.*

Proof. If $k \leqslant R(n)$, then we arbitrarily select $k/2$ edges. In this case, the solution consists of the vertices incident to these edges, adding arbitrarily some vertices, if necessary, in order to have size exactly k. In general, it holds that $\ell \leqslant k(k-1)/2$ and hence $\ell \leqslant R(n)(k-1)/2$. Therefore, the algorithm achieves $R(n)$-approximation ratio in polynomial time.

If $k > R(n)$, then let R^{-1} be the inverse function of R. We consider all possible subgraphs of size k and return the densest one. In this case, the algorithm finds an exact solution with running time $O^*(2^n) = O^*(2^{R^{-1}(k)})$. □

In the two last algorithms, we use again the idea of splitting the vertex set.

Algorithm 7 Decomposition by minimum vertex cover

1: find a minimum vertex cover V^* ($|V^*| = \tau$);
2: consider a partition of V into V_1 and V_2 s.t. $V_1 \subseteq V^*$ and $|V_1| = |V_2 \cap V^*| = \tau/2$;
3: solve DENSEST k-SUBGRAPH on $G[V_1]$ (let A_1 be the solution);
4: solve DENSEST k-SUBGRAPH on $G[V_2]$ (let A_2 be the solution);
5: solve DENSEST k-SUBGRAPH on the bipartite graph $B = (V_1, V_2; E')$ obtained by removing the edges in $E(V_1)$ and $E(V_2)$ (let A_3 be the solution);
6: **return** the best among A_1, A_2 and A_3.

Theorem 10. *Algorithm 7 achieves a $1/3$-approximation ratio for* DENSEST k-SUBGRAPH *in time $O^*(2^{\tau/2})$.*

Proof. By construction $E = E(V_1) \cup E(V_2) \cup E'$. Thus, the approximation ratio of Algorithm 7 is $1/3$, since optimal densest k-subgraphs are built for $G[V_1]$, $G[V_2]$ and B, and one of them contains at least one third of the optimum number of edges. In Line 1, a minimum vertex cover can be computed in $O^*(1.2738^{\tau})$ as in [18]. As $|V_1| = \tau/2$, Line 3 runs in $O^*(2^{\tau/2})$. In Line 4, use GENERIC$(V_2 \cap V^*, V \setminus V^*)$ which, by Proposition 2, runs in $O^*(2^{\tau/2})$, since $|V_2 \cap V^*| = \tau/2$ and $V \setminus V^*$ is an independent set. Finally, as B is a bipartite graph, Line 5 runs in $O^*(2^{\min\{|V_1|,|V_2\}}) = O^*(2^{|V_1|}) = O^*(2^{\tau/2})$. □

Using similar arguments as in the proof of Theorem 10, the following theorem can be proved.

Theorem 11. DENSEST k-SUBGRAPH *is approximable within ratio* $1/2$ *in time* $O^*(2^{n/2})$.

Proof. Consider Algorithm 8. In Lines 3 and 4 of the algorithm the DENSEST i-SUBGRAPH

Algorithm 8 Decompose to equal parts

1: arbitrarily partition V into V_1 and V_2 such that $|V_1| = |V_2| = n/2$;
2: **for** $i = 0$ to k **do**
3: solve DENSEST i-SUBGRAPH on $G[V_1]$ (let $X[i]$ be the solution);
4: solve DENSEST i-SUBGRAPH on $G[V_2]$ (let $Y[i]$ be the solution);
5: build A_1 by determining i that maximizes the edges induced by $A_1 = X[i] \cup Y[k-i]$;
6: solve DENSEST k-SUBGRAPH in the bipartite graph $B = (V_1, V_2; E')$ obtained by removing the edges in $E(V_1)$ and $E(V_2)$ (let A_2 be the solution);
7: **return** the best of A_1 and A_2.

for graphs $G[V_1]$ and $G[V_2]$, respectively, can be computed in $O^*(2^{n/2})$, while Line 5 that finds the best solution for G' is polynomial. Finally, Line 6 runs in $O^*(2^{n/2})$, as B is a bipartite graph. Hence the complexity of the algorithm follows.

As in the proof of Theorem 10 the approximation ratio follows by the facts that $E = E(V_1) \cup E(V_2) \cup E'$, and A_1 and A_2 are optimal for the subgraphs $G' = (V, E \setminus E')$ and B, respectively. □

5 Conclusions

In this paper we studied a well known combinatorial optimization problem with several important applications with respect to two directions. Since there is a large gap concerning the approximability of the DENSEST k-SUBGRAPH problem, we relaxed the polynomial-time complexity assumption of the standard approximation theory by presenting a series of approximation algorithms describing, in general, trade-offs between time complexity and approximation ratio. In a second direction, the goal was to beat the trivial worst-case $O^*(2^n)$ complexity bound for finding an optimal solution for the DENSEST k-SUBGRAPH problem, which was the state-of-the-art upper bound before the conference version of our paper. In this vein, we presented several algorithms based on decomposition or branching schemes and analyses. Note that, the complexity of all the proposed algorithms is either based on the size of the instance or on a specific parameter of it, while our algorithms use polynomial space in contrast with most of the known algorithms in the bibliography. The polynomial-time approximability of the DENSEST k-SUBGRAPH problem on general graphs or even on special and important graph classes, like bipartite graphs, remains the most interesting open question.

Acknowledgements. We would like to thank the anonymous reviewer for her/his valuable comments.

References

[1] R. Andersen and K. Chellapilla. Finding dense subgraphs with size bounds. In *WAW'09*, volume 5427 of *LNCS*, pages 25–36. Springer, 2009.

[2] S. Arora, D. R. Karger, and M. Karpinski. Polynomial time approximation schemes for dense instances of NP-hard problems. *J. Comput. Syst. Sci.*, 58:193–210, 1999.

[3] Y. Asahiro, R. Hassin, and K. Iwama. Complexity of finding dense subgraphs. *Discrete Applied Mathematics*, 121:15–26, 2002.

[4] J. Backer and J. M. Keil. Constant factor approximation algorithms for the densest k-subgraph problem on proper interval graphs and bipartite permutation graphs. *Information Processing Letters*, 110:635–638, 2010.

[5] A. Bhaskara, M. Charikar, E. Chlamtac, U. Feige, and A. Vijayaraghavan. Detecting high log-densities: An $O(n^{1/4})$ approximation for densest k-subgraph. In *STOC'10*, pages 201–210, 2010.

[6] A. Billionnet and F. Roupin. A deterministic approximation algorithm for the densest k-subgraph problem. *International Journal of Operational Research*, 3:301–314, 2008.

[7] A. Björklund, T. Husfeldt, and M. Koivisto. Set partitioning via inclusion-exclusion. *SIAM Journal of Computing*, 39(2):546–563, 2009.

[8] N. Bourgeois, F. Della Croce, B. Escoffier, and V. Th. Paschos. Fast algorithms for MIN INDEPENDENT DOMINATING SET. *Discrete Applied Math.*, 161(4-5):558–572, 2013.

[9] N. Bourgeois, B. Escoffier, and V. Th. Paschos. Approximation of MAX INDEPENDENT SET, MIN VERTEX COVER and related problems by moderately exponential algorithms. *Discrete Applied Mathematics*, 159(17):1954–1970, 2011.

[10] N. Bourgeois, B. Escoffier, V. Th. Paschos, and J. M. M. van Rooij. Fast algorithms for MAX INDEPENDENT SET. *Algorithmica*, 62:382–415, 2012.

[11] N. Bourgeois, A. Giannakos, G. Lucarelli, I. Milis, V. Th. Paschos, and O. Pottié. The MAX QUASI-INDEPENDENT SET PROBLEM. *Journal of Combinatorial Optimization*, 23:94–117, 2012.

[12] N. Bourgeois, A. Giannakos, G. Lucarelli, I. Milis, V. Th. Paschos, Exact and approximation algorithms for DENSEST k-SUBGRAPH. In *WALCOM'13*, volume 7748 of *LNCS*, pages 114–125. Springer-Verlag, 2013.

[13] L. Brankovic and H. Fernau. Combining two worlds: parameterized approximation for vertex cover. In *ISAAC'10*, volume 6506 of *LNCS*, pages 390–402. Spinger-Verlag, 2010.

[14] R. L. Brooks. On coloring the nodes of a network. *Math. Proc. Cambridge Philos. Soc.*, 37:194–197, 1941.

[15] L. Cai. Parameterized complexity of cardinality constrained optimization problems. *The Computer Journal*, 51:102–121, 2007.

[16] L. Cai and X. Huang. Fixed-parameter approximation: conceptual framework and approximability results. In *IWPEC'06*, volume 4169 of *LNCS*, pages 96–108. Springer-Verlag, 2006.

[17] M.-S. Chang, L.-H. Chen, L.-J. Hung, P. Rossmanith, and G.-H. Wu. Exact algorithms for problems related to the densest k-set problem. *Information Processing Letters*, 114:510–513, 2014.

[18] J. Chen, I. A. Kanj, and G. Xia. Improved upper bounds for vertex cover. *Theoretical Computer Science*, 411:3736–3756, 2010.

[19] D. G. Corneil and Y. Perl. Clustering and domination in perfect graphs. *Discrete Applied Mathematics*, 9:27–39, 1984.

[20] M. Cygan, L. Kowalik, and M. Wykurz. Exponential-time approximation of weighted set cover. *Information Processing Letters*, 109(16):957–961, 2009.

[21] M. Cygan and M. Pilipczuk. Exact and approximate bandwidth. *Theoretical Computer Science*, 411(40–42):3701–3713, 2010.

[22] R. G. Downey and M. R. Fellows. *Parameterized complexity*. Monographs in Computer Science. Springer, New York, 1999.

[23] R. G. Downey, M. R. Fellows, and C. McCartin. Parameterized approximation problems. In *IWPEC'06*, volume 4169 of *LNCS*, pages 121–129. Springer-Verlag, 2006.

[24] U. Feige, G. Kortsarz, and D. Peleg. The dense k-subgraph problem. *Algorithmica*, 29:410–421, 2001.

[25] U. Feige and M. Langberg. Approximation algorithms for maximization problems arising in graph partitioning. *Journal of Algorithms*, 41:174-211, 2001.

[26] U. Feige and M. Seltser. On the densest k-subgraph problem. *Technical Report CS97-16*, Weizmann Institute, 1997.

[27] F. V. Fomin, F. Grandoni, and D. Kratsch. A measure & conquer approach for the analysis of exact algorithms. *Journal of the ACM*, 56, 2009.

[28] F. V. Fomin and K. Høie. Pathwidth of cubic graphs and exact algorithms. *Information Processing Letters*, 97:191–196, 2006.

[29] G. Gallo, P. L. Hammer, B. Simeone. Quadratic knapsack problems. *Mathematical Programming*, 12:132-149, 1980.

[30] A. Goldberg. Finding a maximum density subgraph. *Technical Report UCB/CSB 84/171*, Department of Electrical Engineering and Computer Science, University of California, Berkeley, CA, 1984.

[31] G. Jäger and A. Srivastav. Improved approximation algorithms for maximum graph partitioning problems. *Journal of Combinatorial Optimization*, 10:133-167, 2005.

[32] R. M. Karp. Reducibility among combinatorial problems. In R. E. Miller and J. W. Thatcher, editors, *Complexity of computer computations*, pages 85–103. Plenum Press, New York, 1972.

[33] S. Khot. Ruling out PTAS for graph min-bisection, densest subgraph and bipartite clique. In *FOCS'04*, pages 136–145, 2004.

[34] S. Khuller and B. Saha. On finding dense subgraphs. In *ICALP'09*, volume 5555 of *LNCS*, pages 597–608. Springer, 2009.

[35] C. Komusiewicz and M. Sorge. An algorithmic framework for fixed-cardinality optimization in sparse graphs applied to dense subgraph problems. *Discrete Applied Mathematics*, 193:145–161, 2015.

[36] M. Liazi, I. Milis, and V. Zissimopoulos. A constant approximation algorithm for the densest k-subgraph problem on chordal graphs. *Information Processing Letters*, 108:29–32, 2008.

[37] D. Marx. Parameterized complexity and approximation algorithms. *The Computer Journal*, 51(1):60–78, 2008.

[38] H. Moser. *Exact algorithms for generalizations of vertex cover*. PhD thesis, Friedrich-Schiller-Universität Jena, 2005.

[39] T. Nonner. PTAS for densest k-subgraph in interval graphs. *Algorithmica*, 74:528–539, 2016.

[40] U. Pferschy and J. Schauer. Approximation of the quadratic knapsack problem. *INFORMS Journal on Computing*, 28:308-318, 2016.

[41] D. J. Rader Jr. and G. J. Woeginger. The quadratic 0-1 knapsack problem with series-parallel support. *Operations Research Letters*, 30:159–166, 2002.

[42] B. Reed. Paths, stars and the number three. *Combinatorics, Probability and Computing*, 5:277–295, 1996.

[43] R. Taylor. Approximation of the quadratic knapsack problem. *Operations Research Letters*, 44:495–497, 2016.

[44] J. M. M. van Rooij, J. Nederlof, and Th. C. van Dijk. Inclusion/exclusion meets measure and conquer. In *ESA'09*, volume 5757 of *LNCS*, pages 554–565. Springer, 2009.