Laboratoire d'Analyses et Modélisation de Systèmes pour l'Aide à la Décision
UMR 7243

CAHIER DU LAMSADE

324

Juin 2012

Exact and approximation algorithms for DENSEST k-SUBGRAPH

N. Bourgeois, A. Giannakos, G. Lucarelli, I. Milis, V.Th. Paschos

Exact and approximation algorithms for DENSEST k-SUBGRAPH*

N. Bourgeois[(a)] A. Giannakos[(b)] G. Lucarelli[(c)] I. Milis[(d)] V. Th. Paschos[(b)(e)]

June 4, 2012

Abstract

The DENSEST k-SUBGRAPH problem is a generalization of the maximum clique problem, in which we are given a graph G and a positive integer k, and we search among the subsets of k vertices of G one inducing a maximum number of edges. In this paper, we present algorithms for finding exact solutions of DENSEST k-SUBGRAPH improving the standard exponential time complexity of $O^*(2^n)$ and using polynomial space. Two FPT algorithms are also proposed; the first considers as parameter the treewidth of the input graph and uses exponential space, while the second is parameterized by the size of the minimum vertex cover and uses polynomial space. Finally, we propose several approximation algorithms running in moderately exponential or parameterized time.

1 Introduction and preliminaries

In the DENSEST k-SUBGRAPH problem we are given a graph $G = (V, E)$, $|V| = n$, $|E| = m$, 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. DENSEST k-SUBGRAPH is NP-hard, being a generalization of the MAX CLIQUE problem [22]. Moreover, it is NP-complete even to decide if there is a solution with at least $k^{1+\epsilon}$ edges, for any $\epsilon > 0$ [2]. DENSEST k-SUBGRAPH 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.

In this paper, we present (sub)exponential and parameterized algorithms that compute optimal or approximate solutions for the DENSEST k-SUBGRAPH problem. In Section 2 we propose exact algorithms for finding an optimal solution to DENSEST k-SUBGRAPH. These algorithms improve the standard complexity $O^*(2^n)$ for the problem (throughout the paper we use notation $O^*(\cdot)$ that ignores polynomial factors in the complexity expressions). In contrast to the algorithm presented in [11], they need only polynomial space. In this direction, we first present a general decomposition idea which, depending on the way we partition the graph, leads to different time complexities for finding an optimal solution. Next, in Section 2.2, we propose two similar branch-and-cut algorithms and we analyze their complexity using the "measure and conquer" and the "bottom-up" techniques.

*Research supported by the French Agency for Research under the DEFIS program TODO, ANR-09-EMER-010

[(a)] ESSEC, France, nbourgeo@phare.normalesup.org

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

[(c)] Université Pierre et Marie Curie, LIP6, Paris, France, Giorgio.Lucarelli@lip6.fr

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

[(e)] Institut Universitaire de France

In Section 3, we present algorithms of parameterized complexity for DENSEST k-SUBGRAPH. We first propose an algorithm of complexity exponential to the treewidth, tw, of the input graph, supposing that a tree decomposition is given. However, this algorithm uses exponential space. In order to fix this, we show that DENSEST k-SUBGRAPH is FPT with respect to the size τ of a minimum vertex cover of the input graph. Note that $\text{tw} \leqslant \tau$, but the later algorithm uses polynomial space. In Section 4, we first present two **XP**-approximation schemata for DENSEST k-SUBGRAPH whose approximation ratios depend on their complexity (see [16] for a formal definition of the problem-class **XP**). We also give approximation algorithms that run in moderately exponential or parameterized time.

In what follows, 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 a graph G. The diameter $\text{diam}(G)$, of a graph G is the length of the maximum shortest path between any two vertices of the graph. For a graph G, we denote by $\text{tw}(G)$ 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 induced by $G[A]$ and $E(A, B)$ the set of edges with their one endpoint in A and the other in B.

The approximability of DENSEST k-SUBGRAPH has been studied in several papers. For instance, an approximation algorithm achieving ratio $\frac{8k}{9n}$ has been proposed in [4]. In [18], 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 $\epsilon > 0$ [3]. From a negative point of view is known that DENSEST k-SUBGRAPH in general graphs does not admit a PTAS [23].

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 [11]. Notice, however, that this algorithm requires exponential space.

Recall that a problem is fixed-parameter tractable (FPT) with respect to a parameter t if it can be solved (to optimality) with time-complexity $O(f(t)p(n))$ where f is a function that depends on the parameter t and p is a polynomial on the size n of the instance. Cai in [11] proved that DENSEST k-SUBGRAPH is not FPT, i.e., it 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 of the solution ℓ, as any solution cannot contain more than $k(k-1)/2$ edges.

2 Exact algorithms

2.1 A decomposition technique

A general idea for finding an exact solution for the DENSEST k-SUBGRAPH problem in a graph $G = (V, E)$ is to split the vertex set V into two subsets V_1 and V_2. Then, for each i, $1 \leqslant i \leqslant k$, and each subset $A_1 \subseteq V_1$ with $|A_1| = i$, we search for a subset $A_2 \subseteq V_2$, $|A_2| = k - i$, 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 set V_1, as we create all subsets of V_1;
- the complexity of determining, given the set A_1, the set $A_2 \subseteq V_2$.

Hence, it is required for V_1 to be of bounded size and for V_2 to have some specific property that forces A_2 to be determined in polynomial time. We will show that $\Delta(G[V_2]) \leqslant 2$ is such a property.

This idea can be applied to many problems especially to those where feasible solutions are subsets of V satisfying some specific property. As we will see in what follows, this method provides also 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. Therefore, it allows to achieve non-trivial

bounds to running time (using polynomial space), in particular for problems where no bounds better than $O^*(2^n)$ are known.

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 $A_1 \subseteq V_1$, $|A_1| = j$, **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 is 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 $A_1 \subseteq V_1$, $|A_1| \leqslant k$. A solution $A = A_1 \cup A_2$ for the* DENSEST k-SUBGRAPH *problem in G such that $A_2 \subseteq V_2$, $|A_2| = k - |A_1|$, and $|E(A)|$ is maximized, can be found in $O(nk^2)$ time.*

Proof. We will polynomially transform our problem to the quadratic $0-1$ knapsack problem, QKP, [27]. In this problem, we are given a graph $G = (V, E)$ and an integer b. Each vertex $i \in V$ is associated with a cost c_i and a weight w_i and each edge $(i, j) \in E$ has a cost c_{ij}. 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 cost of A, $\sum_{i \in A} c_i + \sum_{i,j \in A} c_{ij}$, is maximized. It has been shown in [27] that this problem can be solved in $O(|V|b^2)$ time for *edge series-parallel* (ESP) graphs. A graph of maximum degree 2 consists of cycles, paths and single vertices and can be converted to a single ESP graph by adding a left and a right fictive terminal vertices and connecting them with the left and the right terminals of each component, respectively (single vertices being both left and right terminals of themselves). In this way, we convert the graph $G[V_2]$ to an ESP graph and we consider each vertex $i \in V_2$ assigned a cost $c_i = |E(\{i\}, A_1)|$ and each edge $(i, j) \in E(V_2)$ assigned a cost $c_{ij} = 1$. The fictive terminals and their incident edges are assigned costs equal to zero. Moreover, each vertex $i \in V_2$ is assigned a weight $w_i = 1$, while fictive terminal vertices have an infinite weight. Finally, the capacity of the knapsack is $k - |A_1|$. An optimal solution to this QKP problem and hence a DENSEST k-SUBGRAPH in G can be found within $O(nk^2)$ time [27]. □

Note that, in the case where $\Delta(G[V_2]) = 0$, i.e., V_2 is an independent set 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 $O^*\left(2^{|V_1|}\right)$ time.*

Proof. In an iteration, let A_1 be the subset of vertices of V_1 that an optimal solution contains. By the optimality of the solution obtained by Proposition 1, the whole solution returned in such an iteration is an optimal one. Line 3 of GENERIC(V_1, V_2) is executed $\sum_{j=0}^{k} \binom{|V_1|}{j}$ times. Hence, the complexity of the algorithm is $O^*\left(2^{|V_1|}\right)$. □

Note that, in the case where $k \leqslant \frac{|V_1|}{2}$ then the term $O^*\left(\binom{|V_1|}{k}\right)$ is a better expression for the complexity.

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 on 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) $O^*\left(2^{\left(1-(5/8)^{\Delta-2}\right)n}\right)$, *if* V_1 *is obtained by repeated excavations of minimum dominating sets;*

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

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

(iv) $O^*\left(2^{n-\mathrm{diam}(G)}\right)$, *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). According to the Reed's theorem [28] for the size of a minimum dominating set: *"any graph $G = (V, E)$ of minimum degree at least three has a dominating set of size at most $3|V|/8$"*. Based on this we propose the following algorithm. The following lemma shows how

Algorithm 1 Decomposition by Minimum Dominating Set

1: $V_\Delta = V$; $D = \emptyset$;
2: **for** $i = \Delta$ to 3 **do**
3: extend $G[V_i]$ to the graph $G'_i = (V'_i, E'_i)$, $V_i \subseteq V'_i$, of minimum order and minimum degree three;
4: find a minimum dominating set $D'_i \subseteq V'_i$ on G'_i;
5: $D_i = D'_i \cap V_i$; $D = D \cup D_i$; $V_{i-1} = V_i \setminus D_i$;
6: **return** GENERIC$(D, V_2 = V \setminus D)$;

to implement Line 3 of the algorithm, without increasing significantly the number of the vertices of the graph.

Lemma 1. *Any graph $G = (V, E)$ can be extended to a graph $G' = (V', E')$ such that $\delta(G') \geqslant 3$ and $|V'| \leqslant |V| + 11$.*

Proof. We consider the vertices of the input graph of degree two or less which appear in connected components that contain at least a vertex of degree three or more. We complete the graph by adding fictive edges between them until they all have degree at least three. Notice that in the following cases this completion will be not successfully finished:

(a) A vertex v of degree one remains. In this case we add the gadget shown in Figure 1(a) and the edges (v, u_1) and (v, u_2). Thus, the number of vertices of the graph becomes $n' = n + 4$.

(b) A vertex v of degree two remains. In this case we add the gadget shown in Figure 1(b), where v coincides with u. Thus, the number of vertices of the graph becomes $n' = n + 5$.

(c) Two adjacent vertices, v_1 and v_2, both of degree two remain. In this case we add the gadget shown in Figure 1(b) and the edges (v_1, u) and (v_2, u). Thus, the number of vertices of the graph becomes $n' = n + 6$.

(d) Two adjacent vertices, v_1 and v_2, of degree one and two, respectively, remain. In this case we add the gadget shown in Figure 1(b), where v_1 coincides with u. Moreover, we add the gadget shown in Figure 1(b) and the edges (v_1, u) and (v_2, u) Thus, the number of vertices of the graph becomes $n' = n + 11$.

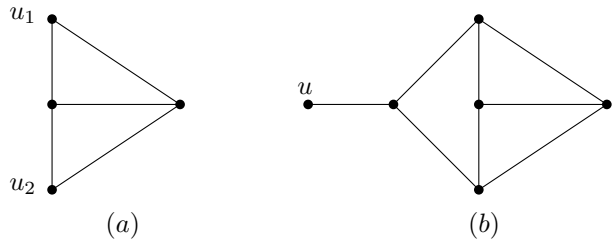

Figure 1:

Note that no vertex of degree three, at the beginning of the process, will receive a new neighbor in this way. Furthermore, the number of vertices n' of the new graph is at most $n + 11$, and hence the proof of the lemma is completed. □

Lemma 2. $|D| \leqslant \left(1 - (5/8)^{\Delta-2}\right) n + o(1)$.

Proof. By Reed's theorem, for the graph G_i' it holds that $|D_i'| \leqslant \frac{3|V_i'|}{8}$. By Lemma 1 we have $|V_i'| \leqslant |V_i| + 11$ and hence $|D_i| = |D_i' \cap V_i| \leqslant |D_i'| \leqslant \frac{3|V_i|}{8} + o(1)$.

Thus, we have $|V_{i-1}| = |V_i| - |D_i| \geqslant |V_i| - \left(\frac{3|V_i|}{8} + o(1)\right) = \frac{5|V_i|}{8} - o(1)$. By solving this recurrence for $i = 3$ we get $|V_2| \geqslant \left(\frac{5}{8}\right)^{\Delta-2} |V_\Delta| - \sum_{j=0}^{\Delta-3} \left(\frac{5}{8}\right)^j \cdot o(1) \geqslant \left(\frac{5}{8}\right)^{\Delta-2} |V| - o(1)$. Hence, it holds that $|D| = |V| - |V_2| \leqslant |V| - \left(\left(\frac{5}{8}\right)^{\Delta-2} |V| - o(1)\right)$, and the proof of the lemma is completed. □

To complete the proof for Item (i), note that $G[V_2] = G[V \setminus D]$ is a graph of maximum degree 2, since $\Delta - 2$ dominating sets have been removed from the initial graph G in Lines 2-5 of the algorithm. The complexity of computing a minimum dominating set in Line 4 of the algorithm is $O^*(1.5048^n)$ [29]. For graphs of maximum degree three, another algorithm is known of complexity $O^*(1.202^n)$ [20]. Finally, according to Lemma 2 the size of the set V_1 that takes as input GENERIC(V_1, V_2) is $|V_1| \leqslant \left(1 - (5/8)^{\Delta-2}\right) n + o(1)$. Using Proposition 2 the proof of Item (i) follows.

Proof of Item (ii). In this case, we search for a minimum vertex cover in the input graph and we compute the complexity of the next algorithm with respect to the chromatic number $\chi(G)$ of the graph. A minimum vertex cover B, can be found in $O^*\left(1.2738^\tau\right) \leqslant O^*\left(1.2738^n\right)$ time [13]. The

Algorithm 2 Decomposition by Minimum Vertex Cover

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

set $V \setminus B$ is a maximum independent set of size at least $\frac{n}{\chi}$, since the vertex set of the input graph can be partitioned into χ independent sets. Hence, by Proposition 2 the first part of Item (ii) of the theorem holds.

If the input graph is a clique or an odd cycle then the DENSEST k-SUBGRAPH problem is polynomial. Otherwise, $\chi \leqslant \Delta$ and the second part of Item (ii) of the theorem holds.

Proof of Item (iii). Consider Algorithm 3. Note that, if there exits a D_i such that $|D_i| \geqslant \frac{n}{\Delta-1}$, then by Line 5 of the algorithm we have that $|D| \leqslant n - \frac{n}{\Delta-1} = \frac{\Delta-2}{\Delta-1}n$. Otherwise, for each i, $3 \leqslant i \leqslant \Delta$, it holds that $|D_i| \leqslant \frac{n}{\Delta-1}$, and hence, $|D| \leqslant (\Delta-2)\frac{n}{\Delta-1}$. Since in both cases $G[V \setminus D]$ is a graph of maximum degree 2, we can apply Proposition 2, completing the proof of Item (iii).

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: **if** $|D_i| \geqslant \frac{n}{\Delta-1}$ **then**
5: $D = V \setminus D_i$; Go to Line 8;
6: **else**
7: $D = D \cup D_i$; $V_{i-1} = V_i \setminus D_i$;
8: **return** GENERIC$(D, V \setminus D)$;

Proof of Item (iv). To prove Item (iv) of the theorem, we propose another algorithm that is obtained by considering the diameter, $\mathrm{diam}(G)$, of the input graph. Note, first, that P contains the vertices

Algorithm 4 Decomposition by the diameter

1: **for** each $v \in V$ **do**
2: create the BFS tree rooted in v;
3: find the longest path, P_v, from v in this BFS;
4: $P = \{P_v : |P_v| \text{ is maximum}, v \in V\}$;
5: **return** GENERIC$(V \setminus P, P)$;

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

The complexity for optimally solving DENSEST k-SUBGRAPH in bipartite graphs can be further improved. Observe that, given a bipartite graph $G = (U, V)$, we can apply GENERIC(U, V) getting an algorithm with running time $O^*\left(\binom{n/2}{k}\right)$ (or $2^{n/2}$ if $k \geqslant n/2$). In the next theorem, we show how to improve this result, by considering the balance of the vertices among the two independent sets in an optimal solution. In what follows, we define $\phi(k, n)$ to be the complexity of our algorithm.

Theorem 2. DENSEST k-SUBGRAPH *can be solved on bipartite graphs in time* $O^*(\phi(k, n))$.

The table below gives an idea of the behavior of $\phi(k, n)$ for different ratios k/n.

k/n	1/100	1/20	1/10	1/6	1/4	1/3
$\phi(k,n)$	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	$\sqrt{2}^n$	$\sqrt{2}^n$

Proof. W.l.o.g., assume that $|U| \leqslant n/2$ and let $\lambda = |U|/n \leqslant 1/2$. GENERIC$(U, V)$ solves DENSEST k-SUBGRAPH in $O^*(\phi(k, \lambda n))$ time, while GENERIC(V, U) solves it in time $O^*(\phi(k, (1-\lambda)n))$. We fix some scalar $\nu(\lambda) \leqslant 1/2$. Notice that either V contains at most νk vertices from an optimal solution, or U contains less than $(1-\nu)k$ of them. Hence, we only need to consider small subsets: $T(n) \leqslant \max_\lambda \min_\nu \{\phi(\nu k, (1-\lambda)n) + \phi((1-\nu)k, \lambda n)\}$. Since the second term in the previous expression involves an increasing and a decreasing function, 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 to give an exact formula, especially considering all the specific cases when k is close to n. As a consequence, we prefer to give a sample of values for the function ϕ. □

2.2 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 complexity. For the analysis of the first algorithm we use the well known technique of *measure and conquer* introduced by Fomin et al. [19]. For the analysis of the second algorithm we use the *bottom-up* technique which has been developed in [8] as a technique for finely measuring the progression of a branching algorithm. This method has led to the best known worst-case complexity for the independent set problem [8], and it has been also used in [9].

Let us first consider a simple branch-and-cut algorithm that branches on a vertex of maximum degree. The branching tree is cut 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 3. *Using measure and conquer, the basic branching algorithm solves* DENSEST k-SUBGRAPH *in time* $O^*\left(2^{\frac{\Delta-1}{\Delta+1}n}\right)$.

Proof. In what follows, we denote by $G_i = (V_i, E_i)$ the subgraph of the input graph G induced by the set of vertices, V_i, not yet examined.

We assign to each vertex $v \in V_i$ a weight according to its degree $\omega(v) = w_{d_{G_i}(v)}$, under the 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.

In each iteration, the degree $d_{G_i}(u)$ for each neighbor $u \in V_i$ of v is decreased by one. Hence, the total weight $W = \sum_{v \in V} w(v)$ is decreased and according to the convexity hypothesis we have:

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

In fact, we only need to verify these inequations for $i \geqslant 3$, since by Proposition 1 the problem is polynomial in graphs of maximum degree 2. However, we are not allowed to fix $w_i = 0$ for $i \leqslant 2$, since we need to verify 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 $w_2 = 2w_3/3$. Indeed the exact value of w_1 has no influence on complexity, while the smallest w_2 the best.

Since all other inequations have to be satisfied, the optimal weight distribution must satisfy:

$$\exists c, \forall 3 \leqslant i \leqslant \Delta, (i+1)w_i - iw_{i-1} = c$$

Summing up these equations, we get $w_i = \frac{(i-2)c+3w_2}{i+1}$. Hence, we have $w_3 = c/2, w_2 = c/3, w_1 = c/6$. Furthermore, we are free to fix $w_\Delta = 1$ (which means $W = n$) and thus we get $c = \frac{\Delta+1}{\Delta-1}$ and, $\forall i \geqslant 2$, $w_i = \frac{(i-1)(\Delta+1)}{(i+1)(\Delta-1)}$.

With this weight function, the recurrence inequality admits as a solution $T(n) \leqslant 2^{\frac{\Delta-1}{\Delta+1}n}$. □

We now slightly modify the previous basic branching algorithm by proceeding to search tree cutting whenever the remaining graph has average degree three. The analysis of this modified branching algorithm is based on the bottom-up method. The following Lemma 3 settles the case where the average degree of the graph is at most 3, while Lemma 4 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 3. 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, choosing each time a vertex of maximum degree, until our graph has maximum degree three. Let n_i, $4 \leqslant i \leqslant \Delta$, be the number of vertices of degree i on which we branch during this step. Moreover, 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 $n - n' < 20n/23$, we greedily branch on vertices of degree three until the graph has maximum degree 2.

(β) If $n - n' \geqslant 20n/23$, we compute a minimum dominating set as described on Algorithm 1 and branch on any vertex of it.

If (α) is true, the running time of our algorithm is $O^*(2^x)$ where:

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

If (β) is true, the running time is $O^*(2^x)$ where:

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

□

Lemma 4. *Assume* DENSEST k-SUBGRAPH *can be computed on graphs with average degree at most* $d-1$, *in time* $O^*(2^{\alpha_d n})$ *for a given* $\alpha_d \geqslant 1/2$, $d \in \mathbb{N}$. *Then, it can be computed on graphs with average degree at least* $d-1$ *in time* $O^*(2^{\alpha_d n + \beta_d(m-(d-1)n/2)})$, *where* $\beta_d = \frac{2(1-\alpha_d)}{d+1}$. *In particular, it can be computed on graphs with average degree at most* d *with running time* $O^*(2^{\alpha_{d+1} n})$, *where* $\alpha_{d+1} = \frac{d\alpha_d + 1}{d+1}$.

Proof. Our hypothesis is true for $m_0 = (d-1)n/2$. Suppose that is true for any pair $n' < n, m' < m$. Since our graph has average degree greater than $d-1$, there exists some vertex of degree d or more. When branching on it, we get:

$$\begin{aligned} T(m,n) &\leqslant 2T(m-d,n-1) \leqslant 2^{1+\alpha_d(n-1)+\beta_d(m-(d-1)n/2)-(d+1)/2} \\ &\leqslant 2^{1-\alpha_d-(d+1)/2\times 2(1-\alpha_d)/(d+1)} \times 2^{\alpha_d n+\beta_d(m-(d-1)n/2)} \leqslant 2^{\alpha_d n+\beta_d(m-(d-1)n/2)} \end{aligned}$$

This remains true by induction for any m, n. □

Theorem 4. DENSEST k-SUBGRAPH *can be solved on graphs of average degree* $\bar{\Delta} \leqslant d$ *with running time* $O^*(2^{\frac{d-27/23}{d+1}n})$*, for any* $d \in \mathbb{N}, d \geqslant 3$*.*

Proof. For $d = 3$ the result follows by Lemma 3. Assume that it is true for $\bar{\Delta} \leqslant d-1$. Then, by Lemma 4, we can find a solution when $\bar{\Delta} \leqslant d$ with running time $O^*(2^{\alpha_{d+1}n})$, where $\alpha_{d+1} = \frac{d\alpha_d+1}{d+1} = \frac{d\frac{d-1-27/23}{d}+1}{d+1} = \frac{d-27/23}{d+1}$. Thus, the statement holds by induction on d. □

3 Parameterized algorithms

Given a graph $G = (V, E)$, a tree decomposition is a pair (X, T), where $X = \{X_1, X_2, \ldots, X_{|X|}\}$, $X_i \subseteq V$, and T is a tree such that: (i) $\bigcup X_i = V$, (ii) for each $e = (u, v) \in E$ there is a X_i where $u, v \in X_i$, and (iii) for each X_i, X_j, X_l such that X_j appears on the path between X_i and X_l it holds that $X_i \cap X_l \subseteq X_j$. The treewidth, tw, of such a decomposition is defined as $\text{tw} = \max\{|X_i|, 1 \leqslant i \leqslant |X|\} - 1$. It is known that finding a minimum treewidth decomposition of a given graph is NP-hard [1]. However, deciding whether there is a tree decomposition of a graph of a fixed treewidth is polynomial [6].

Moser [26], proposes a parameterized algorithm with respect to the treewidth of the input graph for MAX k-COVER. A similar approach can be used for DENSEST k-SUBGRAPH. For the sake of completeness, in the following theorem we briefly describe the algorithm and clarify the issues occurring for DENSEST k-SUBGRAPH.

Theorem 5. *There is a parameterized algorithm for* DENSEST k-SUBGRAPH *with respect to the treewidth that runs in* $O^*(2^{\text{tw}} \cdot k \cdot (\text{tw}^2 + k) \cdot |X|)$ *and uses space exponential to* tw*.*

Proof. Initially, we present an additional definition which is crucial for our analysis. A *nice tree decomposition* is a decomposition where T is a binary tree rooted to a vertex X_r and each vertex X_i of T is of one of the following four types:

- a leaf vertex of $|X_i| = 1$;
- an introduced vertex with one child X_j such that $X_i = X_j \cup \{v\}$ for some vertex $v \in V$;
- a forgotten vertex with one child X_j such that $X_j = X_i \cup \{v\}$ for some vertex $v \in V$;
- a join vertex with two children X_j and X_l such that $X_i = X_j = X_l$.

An algorithm that transforms in linear time a tree decomposition into a nice one of the same treewidth is presented in [24].

Consider now a nice tree decomposition of G and let T_i be the subtree of T rooted at X_i and $G_i = (V_i, E_i)$ be the subgraph of G induced by the vertices in $\bigcup_{X_j \in T_i} X_j$. For each vertex $X_i = (v_1, v_2, \ldots, v_{|X_i|})$ of the tree decomposition, define a configuration vector $c \in \{0,1\}^{|X_i|}$; if

$c[j] = 1$ then $v_j \in X_i$ belongs to the solution for DENSEST k-SUBGRAPH. Moreover, for each vertex X_i, consider a table A_i of size $2^{|X_i|} \times (k+1)$. Each row of A_i represents a configuration and each column represents the number k', $0 \leqslant k' \leqslant k$, of vertices in $V_i \setminus X_i$ included in the solution. The value of an entry of this table equals to the number of edges induced by the solution created for $G[V_i]$. Then $-\infty$ is used to define an infeasible solution.

The algorithm examines the vertices of T in a bottom-up manner and fills in the table A_i for each vertex X_i. In the initialization step, for each leaf vertex X_i and each configuration c we have $A_i[c, k'] = 0$ if $k' = 0$; otherwise $A_i[c, k'] = -\infty$.

If X_i is a forgotten vertex, then consider a configuration c for X_i. In X_j this configuration is extended with the decision whether vertex v is included into the solution or not. Hence, taking into account that $v \in V_i \setminus X_i$ we get $A_i[c, k'] = \max\{A_j[c \times \{0\}, k'], A_j[c \times \{1\}, k' - 1]\}$ for each configuration c and each k', $0 \leqslant k' \leqslant k$.

If X_i is an introduced vertex, then consider a configuration c for X_j. In the case where v is discarded for the solution then $A_i[c \times \{0\}, k'] = A_j[c, k']$. Otherwise, assume that the inclusion of v adds a edges in the solution. Hence, $A_i[c \times \{1\}, k'] = A_j[c, k'] + a$, since k' counts only the vertices of the current solution in $V_i \setminus X_i$.

If X_i is a join vertex, then for each configuration c for X_j and each k', $0 \leqslant k' \leqslant k$, we have to find the best solution obtained by k_j, $0 \leqslant k_j \leqslant k'$, vertices in A_j plus $k' - k_j$ vertices in A_l. However, the number, n_c, of edges in this solution that are induced by X_i are counted twice. Hence, we get $A_i[c, k'] = \max_{0 \leqslant k_j \leqslant k'}\{A_j[c, k_j], A_l[c, k' - k_j]\} - n_c$. □

For the size of the minimum vertex cover τ of the input graph it holds that $\mathrm{tw} \leqslant \tau$. So, Theorem 5 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, which implies only polynomial space. The proof of the following theorem immediately derives by the proof of Theorem 1, Item (ii).

Theorem 6. *There exists an $O^*(2^\tau)$-time algorithm for* DENSEST k-SUBGRAPH *that uses polynomial space.*

We now improve the analysis of Theorem 6 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 7. DENSEST k-SUBGRAPH *can be solved in $O^*(\max\{\gamma^\tau, c^k\})$, for two related constants $\gamma < 2$ and $c > 4$, and needs polynomial space.*

Proof. Note that by the proof of Theorem 6 the running time of GENERIC is $O^*(\sum_{i=1}^{k} \binom{\tau}{i})$. If $\tau \leqslant k$, 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 will prove that, for any $0 < \lambda < 1/2$, we can determine constants $\gamma = \gamma(\lambda) < 2$ and $c = c(\lambda) > 4$ such that DENSEST k-SUBGRAPH can be solved in time $O^*(\max\{\gamma^\tau, c^k\})$. We distinguish the following two cases: $\tau > k \geqslant \lambda\tau$ and $k < \lambda\tau$.

If $k \geqslant \lambda\tau$, then using the fact that $k < \frac{k/\lambda}{2}$ and Stirling's formula we get $\sum_{i=1}^{k} \binom{\tau}{i} \leqslant k\binom{k/\lambda}{k} \sim k\left(\frac{(\lambda^{-1})^{\lambda^{-1}}}{(\lambda^{-1}-1)^{(\lambda^{-1}-1)}}\right)^k = O^*\left(c^k\right)$, for some constant c that depends on λ.

If $k < \lambda\tau$, then $k < \tau/2$ and hence $\sum_{i=1}^{k} \binom{\tau}{i} \leqslant k\binom{\tau}{k} \leqslant k\left(\frac{1}{\lambda^\lambda(1-\lambda)^{(1-\lambda)}}\right)^\tau = O^*\left(\gamma^\tau\right)$, for some constant $\gamma < 2$ that depends on λ. □

The table below contains the values of c and γ 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
$c=\frac{\frac{1}{\lambda^{\frac{1}{\lambda}}}}{(\frac{1}{\lambda}-1)^{(\frac{1}{\lambda}-1)}}$	270.47	53.00	25.81	16.74	12.21	9.48	7.66	6.36	5.38	4.61	4.11
$\gamma=\frac{1}{\lambda^{\lambda}(1-\lambda)^{1-\lambda}}$	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

Under the classic complexity structure hypotheses, there is no constant factor approximation algorithm for DENSEST k-SUBGRAPH that runs in polynomial time. In this section, by relaxing the demand of polynomiality, we present approximation algorithms that run in time exponential, but faster than the time needed for computing an exact solution. This approach has already been considered for several other paradigmatic problems such as MINIMUM SET COVER [14], MIN COLORING [5], MAX INDEPENDENT SET and MIN VERTEX COVER [7], MIN BANDWIDTH [15, 21], etc. Note that, the $O(n^{-(1/4+\epsilon)})$-approximation algorithm with complexity $n^{O(1/\epsilon)}$ presented in [3], 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 [10, 12, 17, 25].

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 either with complexity of the form $O^*(c^n)$, with $1 < c \leqslant 2$, or parameterized complexity.

4.1 XP-approximation algorithms

A general idea to create 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 ρ such that $0 < \rho \leqslant 1$, there is 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 $|E(A_1^*)| \geqslant \frac{\rho}{1-\rho} \cdot |E(A_2^*)|$.*

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

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

that concludes the proof of the proposition. □

Theorem 8. *For any ρ, $0 < \rho \leqslant 1$, Algorithm 5 achieves a ρ-approximation ratio in $O^*(n^{\rho k})$.*

Proof. In an iteration, the algorithm will consider as A 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 created.

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: find the set of vertices $A_2 \in V \setminus A_1$, $|A_2| = (1-\rho)k$, which have the highest degree to A_1;
3: create a solution $A_1 \cup A_2$;
4: **return** the maximum solution found;

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:

$$\begin{aligned}\frac{\text{sol}}{\text{opt}} &\geqslant \frac{|E(A_1^*)| + |E(A_2)| + |E(A_1^*, A_2^*)|}{|E(A_1^*)| + |E(A_2^*)| + |E(A_1^*, A_2^*)|} \\ &\geqslant \frac{|E(A_1^*)| + |E(A_1^*, A_2^*)|}{|E(A_1^*)| + \frac{1-\rho}{\rho}|E(A_1^*)| + |E(A_1^*, A_2^*)|} \geqslant \rho\end{aligned}$$

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 $(1-\rho)k$ arbitrary selected vertices. The following lemma deals with the density of an induced subgraph and will be used to count the number of edges induced by such an optimal DENSEST ρk-SUBGRAPH.

Lemma 5. *Consider a graph $G = (V, E)$ of density $\varpi = 2|E|/|V|(|V|-1)$. For any p, $2 \leqslant p \leqslant |V|$, there exists a set of vertices $V_p \subseteq V$, $|V_p| = p$, such that the induced subgraph $G_p(V_p, E(V_p))$ has density ϖ.*

Proof. Assume for contradiction, that there exist some p's for which the statement of the lemma is false. Let p be the maximum such value. Then, for any $v \in V_{p+1}$:

$$\frac{\varpi \cdot p(p-1)}{2} \geqslant |E(V_{p+1} \setminus \{v\})| + 1$$

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

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

which is a contradiction. □

In the following theorem, we consider 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 9. *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 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 $\mathcal{A}$ and $\varpi' = 2|E(V')|/|V'|(|V'|-1)$ be its density.

Consider an optimal solution $A^* \subseteq V$, $|A^*| = k$, for the DENSEST k-SUBGRAPH problem of density $\varpi^* = 2|E(A^*)|/k(k-1)$. Let $V'' \subseteq A^*$ be a subset of A^* such that $|V''| = \lceil \rho k \rceil + 1$ and $|E(A'')|$ is maximized. Let $\varpi'' = 2|E(V'')|/|V''|(|V''|-1)$ be the density of $G[V'']$. Since $G[V']$ is the DENSEST $(\lceil \rho k \rceil + 1)$-SUBGRAPH and by Proposition 5, it holds that $\varpi' \geqslant \varpi'' \geqslant \varpi^*$, and hence:

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

By completing the solution with $k - \lceil \rho k \rceil - 1$ arbitrary vertices, the theorem follows. □

In Theorem 9, we count only the edges induced by the DENSEST $(\lceil \rho k \rceil + 1)$-SUBGRAPH, as the remaining vertices are selected arbitrarily. In Algorithm 6, we replace this greedy step and we search 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:    create the graph G_{i+1} by removing from G_i the edges of E(A_i);
6:    A = A ∪ A_i;
7:    if the vertices of G_{i+1} consist an independent set then
8:       complete arbitrarily A with vertices in V \ A such that |A| = k;
9:    i = i + 1;
10: return A;
```

Theorem 10. *Let $\mathcal{A}$ be an exact algorithm of complexity $\phi(k,t)$ for finding a* DENSEST k-SUBGRAPH*. For any ρ such that $0 < \rho \leqslant 1$, Algorithm 6 achieves a $\rho(1-\rho/2)$-approximation 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 $|E(A^*)| - |E_i|$ edges, where $|E_i| = \sum_{j=1}^{i} |E(A_j)|$. Thus, 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 $|E_i| \geqslant \rho^2 \left(i - \frac{i(i-1)}{2}\rho^2\right) |E(A^*)|$.

For $i = 1$, by Theorem 9 the inequality holds. Assume that it is true for $i-1$. Then:

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

Let $E(A)$ be the edges of the final solution obtained by the algorithm. As Algorithm 6 iterates at least $1/\rho$ times, we have $|E(A)| \geqslant \rho^2\left(\frac{\frac{1}{\rho}-\left(\frac{1}{\rho}-1\right)}{2\rho}\right)|E\left(A^*\right)| \geqslant \rho\left(1-\frac{\rho}{2}\right)|E\left(A^*\right)|$. □

In general, Algorithm 6 performs better than Algorithm 5 for small values of ρ, since in that case $\rho(1-\rho/2)$ is close to ρ and $\mathcal{A}$ runs faster than exhaustive enumeration. On the other hand, Algorithm 5 outperforms Algorithm 6 when 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 [11], and hence, neither with respect to the size of the solution ℓ. However, in this section we show that there is 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 11. *Let R be any strictly increasing function. There is a $R(n)$-approximation algorithm for* DENSEST k-SUBGRAPH *with complexity parameterized by 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, while we arbitrarily add 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 a $R(n)$-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 of the paper, we use again the idea of decomposing the vertex set.

Algorithm 7 Decomposition by Vertex Cover

1: find a minimum vertex cover V^* ($|V^*| = \tau$);
2: consider a partition of V into V_1 and V_2 such that $V_1 \subseteq V^*$ 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 of A_1, A_2 and A_3;

Theorem 12. *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 opt $/3$ edges. In Line 1, a minimum vertex cover can be computed as in [13]. As $|V_1| = \tau/2$, Line 3 runs in $O^*(2^{\tau/2})$. In Line 4, use GENERIC$(V_2 \cap VC, V \setminus VC)$ which, by Proposition 2, runs in $O^*(2^{\tau/2})$, since $|V_2 \cap VC| = \tau/2$ and $V \setminus VC$ 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 12, the next theorem follows.

Algorithm 8 Decompose to equal parts

1: find an arbitrary partition of 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: find A_1 by determining i that maximizes the edges induced by $A_1 = X[i] \cup Y[k-i]$;
6: 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_2 be the solution);
7: **return** the best of A_1 and A_2;

Theorem 13. *Algorithm 8 achieves a* $1/2$*-approximation ratio for* DENSEST k-SUBGRAPH *in time* $O^*(2^{n/2})$.

Proof. As in the proof of Theorem 12 the approximation ratio follows, by the fact 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.

In Lines 3 and 4 of the algorithm the DENSEST i-SUBGRAPH 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. □

5 Conclusion

We have presented exact, parameterized and moderately exponential approximation algorithms for the DENSEST k-SUBGRAPH problem.

The general idea for creating exact algorithms by partitioning the vertex set as described in GENERIC(V_1, V_2) can be applied to several problems. Note that the complexity of GENERIC(V_1, V_2) is $\max\{O(2^{|V_1|}), T(n)\}$, where $T(n)$ is the complexity of obtaining the vertex set V_1. In all cases considered in Theorem 1 for DENSEST k-SUBGRAPH the term $T(n)$ is smaller than $O(2^{|V_1|})$. However, this might not be true for other kind of partitions or problems.

Moreover, as DENSEST k-SUBGRAPH is not parameterized with respect to k (and hence with ℓ), an interesting question is if there exists a constant factor approximation algorithm of complexity parameterized by k. In our opinion such an algorithm does not exist, since DENSEST k-SUBGRAPH is a generalization of the maximum clique problem, for which it seems to us very unlikely that it admits a fixed-parameter constant factor approximation algorithm.

References

[1] S. Arnborg, D. G. Corneil, and A. Proskurowski. Complexity of finding embeddings in a k-tree. *SIAM Journal on Algebraic and Discrete Methods*, 8:277–284, 1987.

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

[3] 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.

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

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

[6] H. L. Bodlaender. A linear-time algorithm for finding treedecompositions of small treewidth. *SIAM Journal on Computing*, 25:1305–1317, 1996.

[7] 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.

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

[9] 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.

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

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

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

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

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

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

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

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

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

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

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

[21] M. Fürer, S. Gaspers, and S. P. Kasiviswanathan. An exponential time 2-approximation algorithm for bandwidth. In *IWPEC'09*, volume 5917 of *LNCS*, pages 173–184. Springer, 2009.

[22] M. R. Garey and D. S. Johnson. *Computers and intractability: A guide to the theory of NP-completeness*. Freeman, San Francisco, 1979.

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

[24] T. Kloks. *Treewidth, Computations and Approximations*, volume 842 of *LNCS*. Springer, 1994.

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

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

[27] 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.

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

[29] 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.