Computing densest $k$-subgraph with structural parameters

Tesshu Hanaka

Accepted: 20 October 2022 / Published online: 23 December 2022
© The Author(s), under exclusive licence to Springer Science+Business Media, LLC, part of Springer Nature 2022

Abstract

DENSEST $k$-SUBGRAPH is the problem to find a vertex subset $S$ of size $k$ such that the number of edges in the subgraph induced by $S$ is maximized. In this paper, we show that DENSEST $k$-SUBGRAPH is fixed parameter tractable when parameterized by neighborhood diversity, block deletion number, distance-hereditary deletion number, and cograph deletion number, respectively. Furthermore, we give a 2-approximation $2^{tc(G)/2}n^{O(1)}$-time algorithm where $tc(G)$ is the twin cover number of an input graph $G$.

Keywords Densest subgraphs · Sparest subgraphs · Fixed parameter tractability · Structural parameters · Approximation algorithm

1 Introduction

Finding a dense subgraph is an important topic in graph mining. There are many applications for real problems such as community detection in social networks (Dourisboure et al. 2007), spam detection (Gibson et al. 2005), and identification of molecular complexes in protein-protein interaction networks (Bader and Hogue 2003). In this paper, we study the DENSEST $k$-SUBGRAPH problem, which is a graph optimization problem to find a dense structure in a graph. More precisely, the problem is to find a vertex subset $S$ of size $k$ such that the number of edges in the subgraph induced by $S$ is maximized.

The DENSEST $k$-SUBGRAPH problem is NP-hard due to the NP-hardness of $k$-CLIQUE. Thus, several papers study the parameterized complexity of DENSEST $k$-SUBGRAPH. In Bourgeois et al. (2017), Bourgeois et al. show that DENSEST $k$-SUBGRAPH can be solved in time $2^{tw(G)}n^{O(1)}$ and exponential space, and in time $2^{vc(G)}n^{O(1)}$ and polynomial space, respectively, where $tw(G)$ and $vc(G)$ are the
tree-width and the vertex cover number of the input graph \( G \). Here, if a problem has an algorithm with running time \( f(p)n^{O(1)} \) where \( f \) is some computable function, the problem is said to be fixed-parameter tractable (FPT) with respect to \( p \). Therefore, the problem is fixed-parameter tractable when parameterized by tree-width and vertex cover number, respectively. Furthermore, it can be solved in time \( 2^{O(cw(G)\log k)}n^{O(1)} \) where \( cw(G) \) is the clique-width of the input graph (Broersma et al. 2013).

On the other hand, Densest \( k \)-Subgraph is \( \text{W[1]} \)-hard when parameterized by \( k \) due to the \( \text{W[1]} \)-completeness of \( k \)-Clique. Thus, the problem is unlikely to have any algorithm with running time \( f(k)n^{O(1)} \). Also, Densest \( k \)-Subgraph is \( \text{W[1]} \)-hard with respect to clique-width \( cw(G) \) and cannot be solved in time \( 2^{o(cw(G)\log k)}n^{O(1)} \) unless Exponential Time Hypothesis (ETH) fails (Broersma et al. 2013).

In the Densest \( k \)-Subgraph problem, we seek a dense subgraph in a graph. Although the fixed-parameter algorithms with respect to tree-width or vertex cover number are already proposed, such parameters might be large if a graph contains a dense subgraph. In fact, if a graph contains a clique of size \( k \), then the tree-width is at least \( k - 1 \). Thus, it is natural to consider structural parameters for graphs having dense sub-structures.

### 1.1 Our contribution

In this paper, we study the fixed-parameter tractability of Densest \( k \)-Subgraph and Sparsest \( k \)-Subgraph by using structural parameters between clique-width and vertex cover number. Sparsest \( k \)-Subgraph is the dual problem of Densest \( k \)-Subgraph. The problem is to find a vertex subset \( S \) of size \( k \) such that the number of edges in the subgraph induced by \( S \) is minimized. Clearly, Sparsest \( k \)-Subgraph is equivalent to Densest \( k \)-Subgraph on the complement of an input graph. By using two structural parameters: neighborhood diversity and block deletion number, we show the following theorems.

**Theorem 1** Densest \( k \)-Subgraph and Sparsest \( k \)-Subgraph can be solved in time \( f(\text{nd}(G))n^{O(1)} \) where \( \text{nd}(G) \) is the neighborhood diversity of the input graph.

**Theorem 2** Given a block graph deletion set of size \( \text{bd}(G) \), Densest \( k \)-Subgraph and Sparsest \( k \)-Subgraph can be computed in time \( O(2^{\text{bd}(G)}((k^3+\text{bd}(G))n+m)) \).

Theorems 1 and 2 imply that Densest \( k \)-Subgraph and Sparsest \( k \)-Subgraph are fixed-parameter tractable when parameterized by the neighborhood diversity or the block deletion number.

The neighborhood diversity is introduced by Lampis (2012). It is a structural parameter for dense graphs whereas the tree-width is for sparse graphs. Also, the block deletion number of a graph is defined as the minimum number of vertices such that the graph becomes a block graph by removing them. Both parameters are related to other structural parameters between clique-width and vertex cover number. Unlike tree-width, a graph having small neighborhood diversity or small block deletion number may have dense sub-structures. For example, the neighborhood diversity and the block deletion number of a complete graph are 1, respectively, though the tree-width
is \( n - 1 \). Figure 1 shows the relationship between such graph parameters and the parameterized complexity of Densest \( k \)-Subgraph and Sparsest \( k \)-Subgraph.

Then, we generalize the approach of the block deletion number to the bounded clique-width deletion number.

**Theorem 3** Let \( C \) be a class of bounded clique-width graphs. Then given a \( C \)-deletion set \( D \) of \( G \), Densest (Sparsest) \( k \)-Subgraph can be solved in time \( 2^{|D|}n^{O(1)} \).

In particular, cographs and distance-hereditary graphs including block graphs are bounded clique-width graphs, respectively (Courcelle and Olariu 2000; Golumbic and Rotics 2000). Therefore, Theorem 3 implies that Densest (Sparsest) \( k \)-Subgraph is fixed-parameter tractable with respect to distance-hereditary deletion number and cograph deletion number, respectively.

Maximum \( k \)-Vertex Cover is the problem to find a vertex subset \( S \) of size \( k \) that maximizes the number of edges such that at least one its endpoint is in \( S \). For any graph \( G = (V, E) \), \( S \) is an optimal solution of Maximum \( k \)-Vertex Cover in \( G \) if and only if \( V \setminus S \) is an optimal solution of Sparsest \( (n - k) \)-Subgraph in \( G \) (Bougeret et al. 2014). Thus, the following corollary holds.

**Corollary 1** Let \( C \) be a class of bounded clique-width graphs. Maximum \( k \)-Vertex Cover can be solved in time \( f(nd(G))n^{O(1)} \), and \( 2^{|D|}n^{O(1)} \), respectively, where \( nd(G) \) is the neighborhood diversity and \( D \) is a \( C \)-deletion set.

In this paper, we also give an FPT approximation algorithm for the Densest \( k \)-Subgraph problem. In Bourgeois et al. (2017), Bourgeois et al. propose a 3-approximation \( 2^{|vc(G)|/2}n^{O(1)} \)-time algorithm for Densest \( k \)-Subgraph where \( vc(G) \) is the vertex cover number of \( G \). In this paper, we improve the FPT approximation algorithm in Bourgeois et al. (2017). Actually, we give a 2-approximation \( 2^{tc(G)/2}n^{O(1)} \)-time algorithm parameterized by twin cover number \( tc(G) \).

**Theorem 4** There is a 2-approximation \( O(2^{tc(G)/2}((k^3 + tc(G))n + m)) \)-time algorithm for Densest \( k \)-Subgraph where \( tc(G) \) is the twin cover number of an input graph \( G \).

Note that for any graph \( G \), \( tc(G) \leq vc(G) \) holds.

### 1.2 Related work

Densest \( k \)-Subgraph is NP-hard due to the NP-hardness of \( k \)-clique. It remains NP-hard on chordal graphs, comparability graphs, triangle-free graphs and bipartite graphs with maximum degree 3 (Corneil and Perl 1984; Feige and Seltser 1997). On the other hand, it is solvable in polynomial time on cographs and split graphs (Corneil and Perl 1984). For the parameterized complexity, Densest \( k \)-Subgraph is W[1]-hard when parameterized alone by \( k \) or clique-width (Broersma et al. 2013). On the other hand, it is FPT when parameterized by tree-width and clique-width plus \( k \), respectively (Broersma et al. 2013; Bourgeois et al. 2017). On chordal graphs, Densest \( k \)-Subgraph parameterized by \( k \) is fixed-parameter tractable but it does not admit a polynomial kernel unless \( NP \subseteq \text{coNP/poly} \) (Bougeret et al. 2014).
The relationship between graph parameters and the parameterized complexity of DENSEST (SPARSEST) $k$-SUBGRAPH. The parameters $cw, mw, nd, dhd, bd, cod, cd, tc, tw, pw,$ and $vc$ mean clique-width, modular-width, neighborhood diversity, distance-hereditary deletion number, block deletion number, cograph deletion number, cluster deletion number, twin cover number, tree-width, path-width, and vertex cover number, respectively. Connections between two parameters imply that the above one is bounded by some function in the below one. The parameters with underlines are new results in this paper.

The approximability of DENSEST $k$-SUBGRAPH is also well-studied. Kortsarz and Peleg propose an $O(n^{0.3885})$-approximation algorithm (Kortsarz and Peleg 1993). Then, Feige, Kortsarz, and Peleg improve the approximation ratio to $O(n^{\delta})$ for some $\delta < 1/3$ (Feige et al. 2001). The best known approximation algorithm, proposed by Bhaskara et al., archives the approximation ratio $O(n^{1/4+\epsilon})$ within $n^{O(1/\epsilon)}$ time, for any $\epsilon > 0$ (Bhaskara et al. 2010). For the hardness of approximation for DENSEST $k$-SUBGRAPH, even the NP-hardness of approximation for a constant factor is still open. To avoid this situation, using stronger assumptions, several inapproximability results are shown. Khot shows that there is no polynomial-time approximation scheme (PTAS) for DENSEST $k$-SUBGRAPH assuming NP \( \not\subseteq \cap_{\epsilon>0} \text{BPTIME}(2^{n^{\epsilon}}) \) (Khot 2006). Moreover, Manurangsi proves that it does not admit an $n^{1/(\log \log n)^c}$-approximation algorithm for some constant $c > 0$ under Exponential Time Hypothesis (Manurangsi 2017). For the FPT inapproximability of DENSEST $k$-SUBGRAPH, Chalermsook et al. show that there is no $k^{o(1)}$-FPT approximation algorithm assuming Gap-ETH (Chalermsook et al. 2017). Manurangsi, Rubinstein, and Schramm show that there is no $o(k)$-approximation $f(k)n^{O(1)}$-time algorithm for DENSEST $k$-SUBGRAPH under the Strongish Planted Clique Hypothesis (SPCH) (Manurangsi et al. 2021).

SPARSEST $k$-SUBGRAPH is the complementary problem of DENSEST $k$-SUBGRAPH. The problem is NP-hard because it is a generalization of INDEPENDENT SET. As with DENSEST $k$-SUBGRAPH, SPARSEST $k$-SUBGRAPH is fixed-parameter tractable with respect to tree-width. Since for any graph $G$ and its complement $\tilde{G}$, $cw(\tilde{G}) \leq 2cw(G)$ holds, SPARSEST $k$-SUBGRAPH is W[1]-hard when parameterized by clique-width and it cannot be solved in time $2^{o(cw(G) \log k)}n^{O(1)}$ unless ETH fails, whereas it is solvable in time $2^{O(cw(G) \log k)}n^{O(1)}$.

MAXIMUM $k$-VERTEX COVER is NP-hard even on bipartite graphs (Apollonio and Bo 2014; Gwenaël et al. 2015) and W[1]-hard when parameterized by $k$ (Cai 2008; Guo et al. 2007). Using the pipage rounding technique, Ageev and Sviridenko give a 0.75-approximation algorithm (Ageev and Sviridenko 1999). The best known approx-
imation guarantee is 0.929 (Raghavendra and Tan 2012; Manurangsi 2019) and it is tight (Austrin and Stankovic 2019). Although MAXIMUM k-VERTEX COVER is APX-hard (Petrank 1994), there is a \( (1 - \epsilon) \)-approximation algorithm that runs in time \( (1/\epsilon)^k n^{O(1)} \) for any \( \epsilon > 0 \) (Manurangsi 2019).

Independent work

Mizutani and Sullivan independently and simultaneously proved that DENSEST k-SUBGRAPH is fixed-parameter tractable when parameterized by neighborhood diversity and cluster deletion number (Mizutani and Sullivan 2022).

2 Preliminaries

In this paper, we use the standard graph notations. Let \( G = (V(G), E(G)) \) be an undirected graph, where \( V(G) \) and \( E(G) \) denote the set of vertices and the set of edges, respectively. For simplicity, we sometimes denote \( V \) and \( E \) instead of \( V(G) \) and \( E(G) \), respectively. For a vertex subset \( S \), we denote by \( G[S] \) the subgraph induced by \( S \). We call a subgraph with \( \ell \) vertices an \( \ell \)-subgraph. For a vertex \( v \), we denote by \( N(v) \) the set of neighbors of \( v \) and by \( d(v) \) the degree of \( v \). We denote by \( \cdot^T \) the transpose of a vector (resp., matrix). For the basic definitions of parameterized complexity and structural parameters such as tree-width \( tw(G) \), we refer the reader to the book (Cygan et al. 2015).

2.1 Integer quadratic programming

In INTEGER QUADRATIC PROGRAMMING (IQP), the input consists of an \( n \times n \) integer symmetric matrix \( Q \), an \( m \times n \) integer matrix \( A \), an \( l \times n \) integer matrix \( C \), \( m \)-dimensional integer vector \( b \), \( l \)-dimensional integer vector \( d \), and the task is to find an optimal solution \( x \in \mathbb{Z}^n \) of the following optimization problem.

\[
\begin{align*}
\text{max (min)} & \quad x^T Q x + qx \\
\text{subject to} & \quad Ax \leq b, \quad Cx = d, \quad x \in \mathbb{Z}^n.
\end{align*}
\]

Let \( \alpha \) be the largest absolute value of an entry of \( Q, A, C, \) and \( q \). Then, Lokshtanov proved that INTEGER QUADRATIC PROGRAMMING is fixed-parameter tractable when parameterized by \( n \) and \( \alpha \) (Lokshtanov 2015).

**Theorem 5** (Lokshtanov 2015) INTEGER QUADRATIC PROGRAMMING can be solved in time \( f(n, \alpha) L^{O(1)} \) where \( f \) is a computable function and \( L \) is the length of the bit-representation of the input instance.

2.2 Structural parameters

In this subsection, we introduce the definitions of several structural parameters.
2.2.1 Clique-width

We first give the definition of clique-width (see also Courcelle et al. 1993, 2000). A vertex-labeled graph is a graph such that each vertex has an integer as a label. A \( c \)-graph is a vertex-labeled graph with labels \( \{1, \ldots, c\} \). We call a vertex labeled by \( i \) an \( i \)-labeled vertex. Then the clique-width \( \text{cw}(G) \) of \( G \) is defined as the minimum integer \( c \) such that \( G \) can be constructed by the following operations:

- **[O1]** Create a vertex with label \( i \in \{1, 2, \ldots, c\} \);
- **[O2]** Take a disjoint union of two \( c \)-graphs;
- **[O3]** For two labels \( i \) and \( j \), connect every pair of an \( i \)-labeled vertex and \( j \)-labeled vertex by an edge;
- **[O4]** Relabel all the labels of \( i \)-labeled vertices to label \( j \).

For the above operations, we can construct a rooted binary tree, called a \( c \)-expression tree, such that each node corresponds to the above operations. Nodes corresponding to (O1), (O2), (O3), and (O4) are called introduce nodes, union nodes, join nodes, and relabel nodes, respectively. The class of cographs is equivalent to the class of graphs with clique-width at most 2 (Courcelle and Olariu 2000).

2.2.2 Neighborhood diversity

**Definition 1** (Lampis 2012) Two vertices \( u, v \) are called twins if both \( u \) and \( v \) have the same neighbors apart from \( \{u, v\} \). The neighborhood diversity \( \text{nd}(G) \) of a graph \( G \) is the minimum number such that \( V \) can be partitioned into \( \text{nd}(G) \) sets of twin vertices.

We call a set of twin vertices a module. Note that every module forms either a clique or an independent set and two modules either are completely joined by edges or have no edge between them in \( G \). Given a graph, its neighborhood diversity and the corresponding partition can be computed in linear time (Lampis 2012; McConnell and Spinrad 1999; Tedder et al. 2008).

2.2.3 \( G \)-deletion number

For a graph class \( G \), a \( G \)-deletion set \( D \subseteq V \) of \( G \) is a set of vertices such that \( G[V \setminus D] \) is in \( G \). The \( G \)-deletion number \( \partial_G(G) \) of a graph \( G \) is defined as the minimum size of a \( G \)-deletion set in \( G \).

2.2.4 Twin cover and vertex cover

A set of vertices \( X \) is a twin-cover of \( G \) if any edge \( \{u, v\} \) satisfies either (i) \( u \in X \) or \( v \in X \), or (ii) \( N[u] = N[v] \). A vertex cover \( X \) is a set of vertices such that for every edge, at least one endpoint is in \( X \).

2.2.5 Relationship between parameters

For the clique-width \( \text{cw}(G) \), the tree-width \( \text{tw}(G) \), the modular-width \( \text{mw}(G) \), the neighborhood diversity \( \text{nd}(G) \), the block deletion number \( \text{bd}(G) \), the cluster deletion
number $c(d(G))$, the twin cover number $t(c(G))$, and the vertex cover number $v(c(G))$, the following relationship holds.

**Proposition 1** (Bodlaender et al. 1995; Courcelle and Olariu 2000; Gajarský et al. 2013; Ganian 2015; Lampis 2012) For any graph $G$, the following inequalities hold: $cw(G) \leq 2^{vc(G)} + 1$, $tw(G) \leq vc(G)$, $cw(G) \leq mw(G) + 2$, $mw(G) \leq nd(G) \leq 2^{tc(G)} + tc(G)$, and $bd(G) \leq cd(G) \leq tc(G) \leq vc(G)$.

**Proposition 2** Let $C$ be a class of graphs of clique-width at most $c$ and $d_c(G)$ be the $C$-deletion number of a graph $G$. Then for any graph $G$, $cw(G) \leq 2^{d_c(G) + c + 1}$ holds.

**Proof** We generalize the proof of Lemma 4.17 in Sorge et al. (2022). Let $D$ be a minimum $C$-deletion set of $G$. Then the clique-width of $G[V \setminus D]$ is at most $c$. Since the rank-width of a graph is at most the clique-width (Oum and Seymour 2006), the rank-width $rw(G[V \setminus D])$ of $G[V \setminus D]$ is at most $c$. It is known that deleting a vertex decreases the rank-width by at most 1 (Hlinený and Oum S.l, Seese D, Gottlob G, 2007), and hence the rank-width $rw(G)$ of $G$ is at most $c + d_c(G)$. Because $cw(G) \leq 2^{rw(G) + 1}$ holds (Oum and Seymour 2006), we have $cw(G) \leq 2^{d_c(G) + c + 1}$. □

**Corollary 2** For any graph $G$, $cw(G) \leq 2^{bd(G) + 4} \leq 2^{hd(G) + 4}$ holds where $bd(G)$ is the block deletion number and $dhd(G)$ is the distance-hereditary deletion number.

**Proof** The clique-width of a distance-hereditary graph is at most 3 (Golumbic and Rotics 2000). □

### 3 Neighborhood diversity

In this section, we prove Theorem 1. Let $M = \{M_1, \ldots, M_{nd(G)}\}$ be a set of modules with neighborhood diversity $nd(G)$. The *type graph* $Q = (M, E_Q)$ of $G$ is a graph such that each vertex is a module of $G$ and there is an edge between modules if and only if two modules are completely joined in $G$. We denote by $S$ the solution set. Then we define a variable $x_i$ for each module $M_i$. A variable $x_i$ represents the number of vertices in $S \cap M_i$. Moreover, we define $m_i = |M_i|$ for $M_i$. Without loss of generality, we assume that $M_C = \{M_1, \ldots, M_p\}$ is the set of modules that form cliques and $M_f = \{M_{p+1}, \ldots, M_{nd(G)}\}$ is the set of modules that are independent sets.

Then **DENSEST $k$-SUBGRAPH** can be formulated as the following integer quadratic programming problem.

\[
\text{(IQP-DkS)} \quad \begin{align*}
\text{maximize} & \quad \sum_{(M_i, M_j) \in E_Q} x_i x_j + \sum_{i \in \{1, \ldots, p\}} x_i (x_i - 1)/2 \\
\text{subject to} & \quad \sum_{i \in \{1, \ldots, nd(G)\}} x_i = k \\
& \quad 0 \leq x_i \leq m_i \quad \forall i \in \{1, \ldots, nd(G)\} \\
& \quad x_i \in \mathbb{Z} \quad \forall i \in \{1, \ldots, nd(G)\}
\end{align*}
\]

In the objective function of (IQP-DkS), the first term represents the sum of edges between modules in $G[S]$. The second term represents the sum of edges inside modules. Also, the first constraint represents the size constraint.
Without loss of generality, we replace the objective function by:

$$\sum_{\{M_i, M_j\} \in E_Q} 2x_i x_j + \sum_{i \in \{1, \ldots, p\}} x_i (x_i - 1).$$

It is easily seen that (IQP-DkS) can be represented as INTEGER QUADRATIC PROGRAMMING such that the largest absolute value of entries of matrices in (IQP-DkS) is 1. The number of variables in (IQP-DkS) is $n\Delta(G)$ and the number of constraints is $2n\Delta(G) + 1$. By applying Theorem 5, we complete the proof of Theorem 1. Furthermore, the minimization version of (IQP-DkS) is the IQP formulation of SPARSEST k-SUBGRAPH. Thus, SPARSEST k-SUBGRAPH is also fixed-parameter tractable when parameterized by neighborhood diversity.

4 Block deletion number

In this section, we prove Theorem 2. To show this, we define DENSEST (SPARSEST) k-SUBGRAPH WITH WEIGHTED VERTICES as an auxiliary problem.

DENSEST (SPARSEST) k-SUBGRAPH WITH WEIGHTED VERTICES

**Input:** A graph $G = (V, E)$ with vertex weight $w_v$ and an integer $k$.

**Output:** A vertex set $V'$ of size $k$ that maximizes (minimizes) $|\{(u, v) \in E | u, v \in V'\}| + \sum_{v \in V'} w_v$.

Note that if $w_v = 0$ for every $v \in V$, DENSEST (SPARSEST) k-SUBGRAPH WITH WEIGHTED VERTICES is equivalent to DENSEST (SPARSEST) k-SUBGRAPH.

**Lemma 1** Let $G$ be a graph class and $D$ be a $G$-deletion set. If DENSEST (SPARSEST) k-SUBGRAPH WITH WEIGHTED VERTICES can be solved in time $T$, then DENSEST (SPARSEST) k-SUBGRAPH is solvable in time $O(2^{2\Delta(D)}(|D|n + T))$ when $D$ is given. In particular, if $T = (n\Delta(G))^{O(1)}$, it can be solved in time $2^{2\Delta(D)}n^{O(1)}$.

**Proof** Given a $G$-deletion set $D$ of $G$, we first guess $2^{\Delta(D)}$ candidates of partial solutions in $G[D]$. For each candidate $S \subseteq D$, we define the weight $w_v$ of $v \in V \setminus D$ as the number of neighbors in $D$. The weight $w_v$ for $v \in V \setminus D$ means the number of additional edges of the solution when $v$ is added to the solution $S$. Every weight can be computed in time $O(\Delta(D)n)$. Let $k' = k - |S \cap D|$. Then we solve DENSEST (SPARSEST) $k'$-SUBGRAPH WITH WEIGHTED VERTICES with respect to $w_v$ in $G[V \setminus D]$. Thus, the total running time is $O(2^{\Delta(D)}(|D|n + T))$.

In DENSEST (SPARSEST) $k'$-SUBGRAPH WITH WEIGHTED VERTICES with respect to $w_v$ in $G[V \setminus D]$, $w_{\max} = \max_{v \in V \setminus D} w_v$ is bounded by $|D| \leq n$, and hence it can be solved in time $T = (n w_{\max})^{O(1)} = n^{O(1)}$. Therefore, the total running time is $2\Delta(D)n^{O(1)}$.

We then show that DENSEST k-SUBGRAPH WITH WEIGHTED VERTICES can be solved in polynomial time on block graphs.

Suppose that a block graph $G$ is connected. A vertex $v$ is called a cut vertex if $G[V \setminus \{v\}]$ has at least two components. By the definition of a block graph, two blocks

\[ \square \]
Fig. 2 An example of a block graph (left) and its block cut tree (right). In the block-cut tree, the black nodes represent block nodes and the white nodes represent cut nodes.

share exactly one cut vertex. Let \( B = \{ B_1, \ldots, B_\beta \} \) be the set of blocks where \( \beta \) is the number of 2-connected components. For each cut vertex \( v \), we define \( C_v = \{ v \} \). Let \( \mathcal{C} = \{ C_{v_1}, \ldots, C_{v_\gamma} \} \) where \( \gamma \) is the number of cut vertices and let \( \mathcal{X} = B \cup \mathcal{C} \). Then the block-cut tree of \( G \) is a tree \( T = (\mathcal{X}, E) \) where \( E = \{ \{ B_i, C_v \} \mid B_i \cap C_v = \{ v \}, B_i \in B, C_v \in \mathcal{C} \} \) (see Fig. 2). In the block-cut tree, \( B_i \) is called a block node and \( C_v \) is called a cut node. We denote by \( B_r \) the root node in \( T \). The block-cut tree of \( G \) can be computed in linear time (Tarjan 1972).

**Lemma 2** Densest \( k \)-Subgraph with Weighted Vertices on block graphs can be solved in time \( O(k^3n + m) \).

**Proof** Suppose that a block graph \( G \) is connected. For each node \( i \) in a block-cut tree, we define the set of vertices \( V_i \subseteq V \) as the union of all nodes \( B_j \) such that \( j = i \) (if node \( i \) is a block node) or \( j \) is a descendant of \( i \). Note that \( V = V_r \).

Our algorithm is based on dynamic programming on the block-cut tree \( T = (\mathcal{X}, E) \) of \( G \). Except for the root node, we suppose that a block node has its parent cut node \( C_v = \{ v \} \) and we call \( v \in C_v \) the parent cut vertex. Note that the parent node of a block node is a cut node. For \( p \in \{ 0, 1 \} \) and \( \ell \in \{ 0, \ldots, k \} \), let \( A[i, p, \ell] \) be the maximum value of \( \ell \)-subgraphs in \( G[V_i] \). The value \( p \) indicates whether the parent cut vertex is contained in the solution or not. If \( p = 0 \), the parent cut vertex is not contained in the solution, and otherwise, it is contained. In the root node \( B_r \), which is a block node, \( A[r, \ell] \) is defined as the maximum value of \( \ell \)-subgraphs on a block graph. Thus, \( A[r, k] \) is the optimal value of Densest \( k \)-Subgraph with Weighted Vertices. Here, we define \( A[i, p, \ell] = -\infty \) as an invalid case.

In the following, we define the recursive formulas for each node on a block cut tree.

**Leaf block node:** In a leaf block node \( B_i \), let \( v \in B_i \) be the parent cut vertex. Without loss of generality, we suppose \( B_i \setminus \{ v \} = \{ u_1, u_2, \ldots, u_{|B_i|-1} \} \) where \( w_{u_1} \geq w_{u_2} \geq \ldots \geq w_{u_{|B_i|-1}} \). For \( \ell \geq 1 \), we define \( a[\ell] = \sum_{i=1}^{\ell} w_{u_i} \), which is the sum of weights of the top \( \ell \) vertices in the order of weights in \( B_i \setminus \{ v \} \). For \( \ell = 0 \), we set \( a[0] = 0 \).
Then $A[i, p, \ell]$ can be defined as follows:

\[
A[i, p, \ell] = \begin{cases} 
\ell(\ell - 1)/2 + a[\ell] & \text{if } p = 0 \text{ and } 0 \leq \ell \leq |B_i| - 1 \\
\ell(\ell - 1)/2 + a[\ell - 1] + w_v & \text{if } p = 1 \text{ and } 1 \leq \ell \leq |B_i| - 1 \\\n-\infty & \text{otherwise}
\end{cases}
\]

The first case is when the parent cut vertex $v$ is not contained in the solution, and the second case is when $v$ is contained in the solution.

**Internal cut node:** In an internal cut node $C_{v_i} = \{v_i\}$, we compute the maximum value $A[v_i, p, \ell]$ of $\ell$-subgraphs in $G[V_{v_i}]$. If $p = 1$, the solution contains the cut vertex $v_i$, and otherwise it does not. Suppose that the internal cut node $C_{v_i}$ has $t$ subtrees in $T$. Let $T_1, \ldots, T_t$ be the subtrees of $C_{v_i}$ whose root nodes are $C_{v_i}$’s children $B_{j_1}, \ldots, B_{j_t}$.

Using dynamic programming, we compute $A[v_i, p, \ell]$. For $p \in \{0, 1\}$, $s \in \{1, \ldots, t\}$, and $\ell \in \{0, \ldots, k\}$, we compute $c_{v_i}[s, p, \ell]$ that represents the maximum value of $\ell$-subgraphs in the subgraph induced by vertices in the subtrees $T_1, \ldots, T_s$. Note that $A[v_i, p, \ell] = c_{v_i}[t, p, \ell]$. At the base case, we set $c_{v_i}[1, p, \ell] = A[j_1, p, \ell]$ for each $p$ and $\ell$. Then $c_{v_i}[s, p, \ell]$ can be recursively computed by using the following formula:

\[
\begin{align*}
&c_{v_i}[s, p, \ell] = \\
&\begin{cases}
\max \ell' + \ell'' = \ell \{A[j_s, p, \ell'] + c_{v_i}[s - 1, p, \ell'']\} & \text{if } p = 0 \\
\max \ell' + \ell'' = \ell + 1 \{A[j_s, p, \ell'] + c_{v_i}[s - 1, p, \ell'']\} & \text{if } p = 1
\end{cases}
\end{align*}
\]

In the case of $p = 1$, we have to consider the double counting of $v$, and hence we set $\ell' + \ell'' = \ell + 1$. Finally, we set $A[v_i, p, \ell] = c_{v_i}[t, p, \ell]$.

**Internal block node:** In an internal block node $B_i$, let $B_i \setminus \{v\} = \{u_1, \ldots, u_{|B_i| - 1}\}$. Here, for $u \in B_i \setminus \{v\}$ we set:

\[
w'_u[p, \ell] = \begin{cases} 
A[u, p, \ell] & \text{if } u \text{ is a cut vertex} \\
w_u & \text{if } p = 1, \ell = 1, \text{ and } u \text{ is not a cut vertex} \\
0 & \text{if } p = 0 \text{ and } \ell = 0, \text{ and } u \text{ is not a cut vertex} \\
-\infty & \text{otherwise}
\end{cases}
\]

In a sense, $w'_u[p, \ell]$ is the new weight of $u$ taking into account the maximum value of $\ell$-subgraphs in the subtree having $C_u = \{u\}$ as the root.

For $\alpha \in \{0, \ldots, \ell\}$, we compute $a[j, \alpha, \ell]$, which represents the maximum value of $\ell$-subgraphs that contains $\alpha$ vertices among $\{u_1, \ldots, u_j\}$ and $\ell - \alpha$ vertices in the subtrees of $B_i$. Here, for any $j$ and $\ell$, we set $a[j, \alpha, \ell] = -\infty$ if $\alpha = -1$. For $j = 1$, we set:

\[
a[j, \alpha, \ell] = \begin{cases} 
w'_u[0, \ell] & \text{if } \alpha = 0 \\
w'_u[1, \ell] & \text{if } \alpha = 1 \\
-\infty & \text{otherwise}
\end{cases}
\]
The first case is when \( u_1 \) is not contained in the solution and the second case is when \( u_1 \) is in the solution.

For \( j \in \{2, \ldots, |B_i| - 1\} \) and \( 0 \leq \alpha \leq \ell \), we compute \( a[j, \alpha, \ell] \) by using the following recursive formula:

\[
a[j, \alpha, \ell] = \max_{\ell' + \ell'' = \ell} \max \left\{ a[j - 1, \alpha, \ell'] + w'_{uj}[0, \ell''], \quad a[j - 1, \alpha - 1, \ell'] + w'_{uj}[1, \ell''] \right\}.
\]

The first case is when \( u_j \) is not contained in the solution and the second case is when \( u_j \) is in the solution. After the calculation, we obtain \( a[|B_i| - 1, \alpha, \ell] \) for every \( \alpha \) and \( \ell \). Let \( a[\alpha, \ell] = a[|B_i| - 1, \alpha, \ell] \). Finally, the maximum value of \( \ell \)-subgraphs on a node \( B_i \) can be computed as follows:

\[
A[i, p, \ell] = \begin{cases} 
\max_{0 \leq \alpha \leq \ell} \{ a[\alpha, \ell] + \alpha(\alpha - 1)/2 \} & \text{if } p = 0 \\
\max_{0 \leq \alpha \leq \ell - 1} \{ a[\alpha, \ell - 1] + \alpha(\alpha + 1)/2 \} + w_v & \text{else}
\end{cases}
\]

The first case is when the parent cut vertex \( v \) is not contained in the solution and the second case is when \( v \) is in the solution. Note that \( G[B_i] \) forms a clique.

**Root node:** In the root node \( X_r \), we compute \( a[r, \alpha, \ell] \) that represents the maximum value of \( \ell \)-subgraphs that contains \( \alpha \) vertices in \( B_r \) and \( \ell - \alpha \) vertices in the subtrees of \( B_r \) by the same way as internal block nodes. Then, we set \( A[r, \ell] = \max_{0 \leq \alpha \leq \ell} \{ a[r, \alpha, \ell] + \alpha(\alpha - 1)/2 \} \) as the maximum value of \( \ell \)-subgraph in \( G = G[V_r] \). Finally, we obtain the maximum value \( A[r, k] \) of \( k \)-subgraphs on \( G \).

In the algorithm, we first compute the block-cut tree of \( G \) in linear time (Tarjan 1972). Since there are at most \( n \) nodes in a block graph, the number of \( A[i, p, \ell]'s \) is \( O(kn) \). For \( a[j, \alpha, \ell] \), each \( j \) corresponds to a vertex in a node and it appears exactly once. Thus, the number of \( a[j, \alpha, \ell]'s \) is \( O(k^2n) \). For each \( u \in B_i \setminus \{v\} \), \( w'_{u}[p, \ell] \) is defined. Thus, the number of \( w'_{u}[p, \ell]'s \) is \( O(kn) \). Finally, the number of children nodes for all \( C_v \)'s is at most the number of block nodes. Therefore, the total number of \( c_v[j, p, \ell]'s \) for all \( C_v \)'s is bounded by \( O(kn) \).

Since for fixed \( p, \ell \), and \( \alpha \), \( A[i, p, \ell], a[j, \alpha, \ell], c_u[j, p, \ell], \) and \( w'_u[p, \ell] \) can be computed in time \( O(k) \), we can compute \( A[r, k] \) in time \( O(k^3n) \).

It is easily seen that the algorithm can be applied to the case that \( G \) is not connected. We first compute the optimal value of DENSEST \( \ell \)-SUBGRAPH WITH WEIGHTED VERTEICES for every \( \ell \) on each connected component of \( G \). Then we again use dynamic programming. Let \( G_1, \ldots, G_t \) be the connected components of \( G \). For \( i \in \{1, \ldots, t\} \), we denote by \( A_i[r, \ell] \) the optimal value of DENSEST \( \ell \)-SUBGRAPH WITH WEIGHTED VERTEICES on \( G_i \). Let \( B[i, \ell] \) be the maximum value of \( \ell \)-subgraphs on the union of \( G_1, \ldots, G_i \). At the base case, we set \( B[1, \ell] = A_1[r, \ell] \) for each \( 0 \leq \ell \leq k \). Then for each \( i \in \{1, \ldots, t\} \) and \( \ell \in \{0, \ldots, k\} \), we set \( B[i, \ell] = \max_{0 \leq \ell' \leq \ell} \{ B[i - 1, \ell'] + A_i[r, \ell - \ell'] \} \). Clearly, \( B[t, \ell] \) is the maximum value of \( \ell \)-subgraphs on \( G \) and it can be computed in time \( O(kt) \). Since \( r \leq n \), the total running time is \( O(k^3n + m) \). \( \square \)
By using Lemma 2, we complete the proof of Theorem 2.

Since Block Vertex Deletion and Cluster Vertex Deletion can be computed in time $4^{bd(G)}n^{O(1)}$ (Agrawal et al. 2016) and $O(1.9102^{cd(G)}cd(G)(n + m))$ (Boral et al. 2016), respectively, we also obtain the following corollary.

**Corollary 3** Densest $k$-Subgraph and Sparsest $k$-Subgraph can be solved in time $4^{bd(G)}n^{O(1)}$ and $O(2^{cd(G)}cd(G)(k^3n + m))$ where $bd(G)$ and $cd(G)$ are the block deletion number and the cluster deletion number of an input graph, respectively.

## 5 Generalization to bounded clique-width deletion number

In this section, we generalize the results in Sect. 4. Let $\mathcal{C}$ be a class of bounded clique-width graphs. Because the clique-width of a block graph is at most 3, $\mathcal{C}$ includes block graphs.

We show that Densest (Sparsest) $k$-Subgraph with Weighted Vertices can be solved in time $(k + 1)^{2c}w_{\text{max}}n^{O(1)}$ on bounded clique-width graphs.

**Lemma 3** Given a $c$-expression tree $T$, Densest (Sparsest) $k$-Subgraph with Weighted Vertices can be solved in time $(k + 1)^{2c}w_{\text{max}}n^{O(1)}$.

**Proof** We give an algorithm based on dynamic programming by modifying the algorithm proposed in Theorem 1 in Broersma et al. (2013).

For a node $t$ in $T$, we denote by $G_t = (V_t, E_t)$ the vertex-labeled subgraph of $G$ represented by node $t$. Then, we define a DP table for each node $t$ which stores $c + 2$ positive integers $\ell$, $W$, and $s_1, \ldots, s_c$.

For each row in a DP table of node $t$, we define $D_t[\ell, W, s_1, s_2, \ldots, s_c] \in \{\text{true, false}\}$ such that $D_t[\ell, W, s_1, s_2, \ldots, s_c] = \text{true}$ if and only if there is a set $S$ in $G_t$ such that:

- for every label $i$, $s_i$ is the number of $i$-labeled vertices in $S$,
- $|S| = \sum_{i \in \{1, \ldots, c\}} s_i = \ell$, and
- $\sum_{i \in \{1, \ldots, c\}} w_i + |E_t(G[V_t \cap S])| \geq W$.

Note that $0 \leq \ell \leq k$, $0 \leq W \leq nw_{\text{max}} + n(n - 1)/2$, and $0 \leq s_i \leq k$ for each $i$. The size of a DP table in a node is at most $(nw_{\text{max}} + n^2) \cdot (k + 1)^{c+1} = (k + 1)^{c+1}w_{\text{max}}n^2$. In the root node $r$ of a $c$-expression tree, a row maximizing $W$ such that $D_t[k, W, s_1, s_2, \ldots, s_c] = \text{true}$ in the DP table indicates an optimal solution.

**Introduce node:** In an introduce node $t$, suppose that an $i$-labeled vertex $v$ is introduced. Then $D_t[\ell, W, s_1, s_2, \ldots, s_c] = \text{true}$ if and only if

- $\ell = 1$, $W = w_v$, $s_i = 1$, and $s_l = 0$ for $l \neq i$, or
- $\ell = 0$, $s_i = 0$ for every $l \in \{1, \ldots, c\}$.

In the former case, $v$ is contained in $S$, and in the latter case $v$ is not contained in $S$. Note that $G_t$ consists of exactly one $i$-vertex $v$.

**Union node:** In a union node $t$, let $t_1$ and $t_2$ be its children nodes. Then, $D_t[\ell, W, s_1, s_2, \ldots, s_c] = \text{true}$ if and only if there is a pair of tuples $(\ell_1, W_1, s_1^1, s_2^1, \ldots, s_c^1)$ in node $t_1$ and $(\ell_2, W_2, s_1^2, s_2^2, \ldots, s_c^2)$ in node $t_2$ such that:

$\square$ Springer
- \( D_t[\ell_1, W_1, s_1^1, s_2^1, \ldots, s_c^1] = \text{true} \),
- \( D_t[\ell_2, W_2, s_1^2, s_2^2, \ldots, s_c^2] = \text{true} \),
- \( \ell_1 + \ell_2 = \ell \),
- \( W_1 + W_2 = W \), and
- \( s_l^1 + s_l^2 = s_l \) for every \( l \in \{1, \ldots, c\} \).

**Join node:** In a join node \( t \), let \( t' \) be its child node. Because all \( i \)-labeled vertices and all \( j \)-labeled vertices are joined by adding all possible edges between them, we define \( D_t[\ell, W, s_1, s_2, \ldots, s_c] = \text{true} \) if and only if there is a tuple \((\ell', W', s_1', s_2', \ldots, s_c')\) in node \( t' \) such that:
- \( D_{t'}[\ell', W', s_1', s_2', \ldots, s_c'] = \text{true} \),
- \( \ell' = \ell \),
- \( W = W' + s_i^l s_j^l \), and
- \( s_l' = s_l \) for every \( l \in \{1, \ldots, c\} \).

**Relabel node:** In a relabel node \( t \), label \( i \) is changed to label \( j \). Thus, we define \( D_t[\ell, W, s_1, s_2, \ldots, s_c] = \text{true} \) if and only if \( s_i = 0 \) and there is a tuple \((\ell', W', s_1', s_2', \ldots, s_c')\) in its child node \( t' \) such that:
- \( D_{t'}[\ell', W', s_1', s_2', \ldots, s_c'] = \text{true} \),
- \( \ell' = \ell \),
- \( W = W' \),
- \( s_i' + s_j' = s_j \), and
- \( s_l' = s_l \) for every \( l \in \{1, \ldots, c\} \setminus \{i, j\} \).

It is easily seen that each DP table can be computed in polynomial time. Because the size of a DP table is at most \((k + 1)^c + 1\) \( w_{\text{max}} n^2 \), the total running time is \((k + 1)^{2c} w_{\text{max}} n^{O(1)} \). \(\square\)

One can compute a \((2^{c w(G)} + 1 - 1)\)-expression tree of a graph with clique-width \( c w(G) \) in time \( O(n^3) \) (Hlinený and Oum 2008; Oum 2008; Oum and Seymour 2006). Thus, DENSEST (SPARSEST) \( k \)-SUBGRAPH WITH WEIGHTED VERTICES on bounded clique-width graphs can be computed in time \( w_{\text{max}} n^{O(1)} \). Therefore, we immediately obtain Theorem 3 by Lemma 1.

### 6 FPT approximation algorithm

In this section, we prove Theorem 4. Actually, we give a 2-approximation \( 2^{\text{bd}(G)} n^{O(1)} \)-time algorithm for DENSEST \( k \)-SUBGRAPH.

**Theorem 6** Given a block deletion set of size \( \text{bd}(G) \), there is a 2-approximation algorithm for DENSEST \( k \)-SUBGRAPH that runs in time \( O(2^{\text{bd}(G)} / (k^3 + \text{bd}(G)) n + m)) \).

Our algorithm is based on an FPT approximation algorithm proposed in Bourgeois et al. (2017). Given a block deletion set \( D \) of size \( \text{bd}(G) \), let \( V_1 \subseteq D \) be a subset of \( D \)
of size \( \lceil \text{bd}(G)/2 \rceil \) and let \( V_2 = V \setminus V_1 \) (see Fig. 3). Note that \( |V_2 \cap D| = \lceil \text{bd}(G)/2 \rceil \).

Let \( E_1 \) and \( E_2 \) be the set of edges in \( G[V_1] \) and \( G[V_2] \), respectively. Also, let \( E' = E \setminus (E_1 \cup E_2) \).

We solve \textsc{Densest }\( k \)-\textsc{Subgraph} for two graphs \( G' = (V_1 \cup V_2, E') \) and \( G'' = (V_1 \cup V_2, E_1 \cup E_2) \). Note that \( E' \) and \( E_1 \cup E_2 \) are disjoint and \( G' \) is bipartite. First, we solve \textsc{Densest }\( k \)-\textsc{Subgraph} in \( G'' \). To do this, we solve \textsc{Densest }\( i \)-\textsc{Subgraph} for every \( i \in \{0, \ldots, k\} \) in \( G[V_1] \) and \( G[V_2] \), respectively. Let \( S_i \) and \( s_i \) be the optimal solution and its value of \textsc{Densest }\( i \)-\textsc{Subgraph} in \( G[V_j] \) for \( j \in \{1, 2\} \). We can observe that an optimal solution in \( G'' = (V_1 \cup V_2, E_1 \cup E_2) \) is \( S'_i \cup S''_i \) such that \( i_1 + i_2 = k \) and \( s'_i + s''_i \) is maximized because \( G'' \) consists of two disjoint graphs \( G[V_1] \) and \( G[V_2] \). Also, we solve \textsc{Densest }\( k \)-\textsc{Subgraph} on \( G' \). Let \( S' \) and \( S'' \) be optimal solutions in \( G' \) and \( S'' \), respectively. Finally, we output the larger of \( S' \) and \( S'' \).

Since one of \( G[S'] \) and \( G[S''] \) has at least half of the optimal number of edges, the approximation ratio of this algorithm is 2.

Finally, we estimate the running time. Because \( |V_1| = \lceil \text{bd}(G)/2 \rceil \) and the block deletion number of \( G[V_2] \) is \( \lceil \text{bd}(G)/2 \rceil \), we can compute \textsc{Densest }\( k \)-\textsc{Subgraph} in \( G[V_1] \) and \( G[V_2] \) in time \( O(2^{\text{bd}(G)/2}((k^3 + \text{bd}(G))n + m)) \), respectively. Thus, \( S'' \) can be computed in time \( O(2^{\text{bd}(G)/2}((k^3 + \text{bd}(G))n + m)) + O(k) = O(2^{\text{bd}(G)/2}((k^3 + \text{bd}(G))n + m)) \). Furthermore, since \( G' \) is a bipartite graph for \( V_1 \) and \( V_2 \), \( V_1 \) is a vertex cover, and hence it is a block deletion set. Thus, \textsc{Densest }\( k \)-\textsc{Subgraph} in \( G' \) can be computed in time \( O(2^{\text{bd}(G)/2}((k^3 + \text{bd}(G))n + m)) \). Consequently, the total running time of this algorithm is \( O(2^{\text{bd}(G)/2}((k^3 + \text{bd}(G))n + m)) \). \hfill \Box

We can compute a minimum twin cover in time \( O(1.2738^{\text{tc}(G)} + \text{tc}(G)n + m) \) (Ganian 2015). Since \( \text{bd}(G) \leq \text{tc}(G) \) and \( 1.2738 < \sqrt{2} \), we obtain Theorem 4.

7 Conclusion

In this paper, we studied the fixed-parameter tractability of \textsc{Densest }\( k \)-\textsc{Subgraph} by using structural parameters between clique-width and vertex cover. We showed that \textsc{Densest }\( k \)-\textsc{Subgraph} is fixed-parameter tractable parameterized by neighborhood diversity, block deletion number, distance-hereditary deletion number, and cograph deletion number, respectively. These results also hold for \textsc{Sparsest }\( k \)-\textsc{Subgraph} and \textsc{Maximum }\( k \)-\textsc{Vertex Cover}. Moreover, we designed a 2-approximation \( 2^{\text{tc}(G)/2}n^{O(1)} \)-time algorithm for \textsc{Densest }\( k \)-\textsc{Subgraph}. This
improved a 3-approximation \(2^{\frac{\text{vc}(G)}{2}n^{O(1)}}\)-time algorithm proposed in Bourgeois et al. (2017).

As for future work, it is worth investigating the parameterized complexity for other structural parameters. In particular, one of the most notable open questions would be the parameterized complexity for modular-width. Also, it might be interesting to discuss whether there is a faster algorithm parameterized by neighborhood diversity without going through quadratic integer program.

**Funding** This work is partially supported by JSPS KAKENHI Grant Numbers JP19K21537, JP21H05852, JP21K17707, JP22H00513.

**Data Availability** No data is associated to this manuscript.

**Declarations**

**Conflict of interest** The authors have not disclosed any competing interests.

**References**

Ageev AA, Sviridenko MI (1999) Approximation algorithms for maximum coverage and max cut with given sizes of parts. In: Proceedings of the 7th international conference on integer programming and combinatorial optimization (IPCO 1999), pp 17–30

Agrawal A, Kolay S, Lokshakov D, Saurabh S (2016) A faster FPT algorithm and a smaller kernel for block graph vertex deletion. In: Proceedings of the 12th Latin American theoretical informatics symposium (LATIN 2016), pp 1–13

Apollonio N, Bo S (2014) The maximum vertex coverage problem on bipartite graphs. Discret Appl Math 165:37–48

Austrin P, Stankovic A (2019) Global cardinality constraints make approximating some max-2-csps harder. In: Proceedings of the 22nd international conference on approximation algorithms for combinatorial optimization problems and the 23rd international conference on randomization and computation (APPROX/RANDOM 2019), pp 24:1–24:17

Bader GD, Hogue CW (2003) An automated method for finding molecular complexes in large protein interaction networks. BMC Bioinf 4:1–27

Bhaskara A, Charikar M, Chlamtac E, Feige U, Vijayaraghavan A (2010) Detecting high log-densities: an \(O(n^{1.5})\) approximation for densest \(k\)-subgraph. In: Proceedings of the 42th ACM symposium on theory of computing (STOC 2010)

Bodlaender HL, Gilbert JR, Hafsteinsson H, Kloks T (1995) Approximating treewidth, pathwidth, frontsize, and shortest elimination tree. J Algorithms 18(2):238–255

Boral A, Cygan M, Kociumaka T, Pilipczuk M (2016) A fast branching algorithm for cluster vertex deletion. Theory Comput Syst 58(2):357–376

Bougeret M, Bousquet N, Giroudeau R, Wattrignant R (2014) Parameterized complexity of the sparsest \(k\)-subgraph problem in chordal graphs. In: Proceedings of the 40th international conference on current trends in theory and practice of computer science (SOFSEM 2014), pp 150–161

Bourgeois N, Giannakos A, Lucarelli G, Milis I, Paschos VT (2017) Exact and superpolynomial approximation algorithms for the densest \(k\)-subgraph problem. Eur J Oper Res 262(3):894–903

Broersma H, Golovach PA, Patel V (2013) Tight complexity bounds for fpt subgraph problems parameterized by the clique-width. Theoret Comput Sci 485:69–84

Cai L (2008) Parameterized complexity of cardinality constrained optimization problems. Comput J 51(1):102–121

Chalermsook P, Cygan M, Kortsarz G, Laekhanukit B, Manurangsi P, Nanongkai D, Trevisan L (2017) From Gap-ETH to FPT-inapproximability: clique, dominating set, and more. In: Proceedings of the 58th annual IEEE symposium on foundations of computer science (FOCS 2017), pp 743–754
Corneil DG, Perl Y (1984) Clustering and domination in perfect graphs. Discret Appl Math 9(1):27–39
Courcelle B, Olariu S (2000) Upper bounds to the clique width of graphs. Discret Appl Math 101(1):77–114
Courcelle B, Engelfriet J, Rozenberg G (1993) Handle-rewriting hypergraph grammars. J Comput Syst Sci 46(2):218–270
Courcelle B, Makowsky JA, Rotics U (2000) Linear time solvable optimization problems on graphs of bounded clique-width. Theory Comput Syst 33(2):125–150
Cyg an M, Fomin FV, Kowalik Ł, Lokshatanov D, Marx D, Pilipczuk M, Pilipczuk M, Saurabh S (2015) Parameterized algorithms. Springer, Berlin
Dourisboure Y, Geraci F, Pellegrini M (2007) Extraction and classification of dense communities in the web. In: Proceedings of the 26th international world wide web conference (WWW 2017), pp 461–470
Feige U, Seltser M (1997) On the densest \( k \)-subgraph problem. Technical report, Department of Applied Mathematics and Computer Science, The Weizmann Institute, Rehobot
Feige U, Peleg D, Kortsarz G (2001) The dense \( k \)-subgraph problem. Algorithmica 29(3):410–421
Gajarský J, Lampis M, Ordyniak S (2013) Parameterized algorithms for modular-width. In: Proceedings of the 8th international symposium on parameterized and exact computation (IPEC 2013), pp 163–176
Ganian R (2015) Improving vertex cover as a graph parameter. Discrete Math Theor Comput Sci 17(2):77–100
Gibson D, Kumar R, Tomkies A (2005) Discovering large dense subgraphs in massive graphs. In: Proceedings of the 31st international conference on very large data bases (VLDB 2005), pp 721–732
Golumbic M, Rotics U (2000) On the clique-width of some perfect graph classes. Int J Found Comput Sci 11(03):423–443
Guo J, Niedermeier R, Wernicke S (2007) Parameterized complexity of vertex cover variants. Theory Comput Syst 41(3):501–520
Gwenaël J, Adrian V (2015) Reducing the rank of a matroid. Discrete Math Theor Comput Sci 17(2):143–156
Hlinený P, Oum S (2008) Finding branch-decompositions and rank-decompositions. SIAM J Comput 38(3):1012–1032
Hlinený P, SI Oum, Seese D, Gottlob G (2007) Width parameters beyond tree-width and their applications. Comput J 51(3):326–362
Khot S (2006) Ruling out ptas for graph min-bisection, dense \( k \)-subgraph, and bipartite clique. SIAM J Comput 36(4):1025–1071
Kortsarz G, Peleg D (1993) On choosing a dense subgraph. In: Proceedings of the 34th IEEE symposium on foundations of computer science (FOCS1993), pp 692–701
Lampis M (2012) Algorithmic meta-theorems for restrictions of treewidth. Algorithmica 64(1):19–37
Lokshatanov D (2015) Parameterized integer quadratic programming: variables and coefficients. CoRR, abs/1511.00310
Manurangsi P (2017) Almost-polynomial ratio ETH-hardness of approximating densest \( k \)-subgraph. In: Proceedings of the 49th annual ACM SIGACT symposium on theory of computing (STOC 2017), pp 954–961
Manurangsi P (2019) A note on max k-vertex cover: faster fpt-as, smaller approximate kernel and improved approximation. In: Proceedings of the 2nd symposium on simplicity in algorithms (SOSA 2019), vol 69, pp 15:1–15:21
Manurangsi P, Rubinstein A, Schramm T (2021) The strongish planted clique hypothesis and its consequences. In: Proceedings of the 12th innovations in theoretical computer science conference (ITCS 2021), vol 185, pp 10:1–10:21
McConnell RM, Spinrad JP (1999) Modular decomposition and transitive orientation. Discret Math 201(1):189–241
Mizutani Y, Sullivan BD (2022) Improved parameterized complexity of happy set problems
Oum S (2008) Approximating rank-width and clique-width quickly. ACM Trans Algorithms 5(1):10:1-10:20
Oum S, Seymour P (2006) Approximating clique-width and branch-width. J Combin Theory Ser B 96(4):514–528
Petrank E (1994) The hardness of approximation: gap location. Comput Complex 4(2):133–157
Raghavendra P, Tan N (2012) Approximating csps with global cardinality constraints using SDP hierarchies. In: Rabani Y (ed) Proceedings of the twenty-third annual ACM-SIAM symposium on discrete algorithms, SODA 2012, Kyoto, Japan, January 17–19, 2012. SIAM, pp 373–387
Sorge M, Weller M, Foucaud F, Suchý O, Ochem P, Vatshelle M, Woeginger GJ (2022) The graph parameter hierarchy. Manuscript, 2019. https://manyu.pro/assets/parameter-hierarchy.pdf. Accessed 30 Oct

Tarjan R (1972) Depth-first search and linear graph algorithms. SIAM J Comput 1(2):146–160

Tedder M, Corneil D, Habib M, Paul C (2008) Simpler linear-time modular decomposition via recursive factorizing permutations. In: Proceedings of the 35th international colloquium on automata, languages and programming (ICALP 2008), pp 634–645

Publisher’s Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Springer Nature or its licensor (e.g. a society or other partner) holds exclusive rights to this article under a publishing agreement with the author(s) or other rightsholder(s); author self-archiving of the accepted manuscript version of this article is solely governed by the terms of such publishing agreement and applicable law.