Single-exponential FPT algorithms for enumerating secluded F-free subgraphs and deleting to scattered graph classes

Journal of Computer and System Sciences 148 (2025) 103597

Contents lists available at ScienceDirect

Journal of Computer and System Sciences

journal homepage: www.elsevier.com/locate/jcss

Single-exponential FPT algorithms for enumerating secluded $\mathcal{F}$-free subgraphs and deleting to scattered graph classes ☆

Bart M.P. Jansen [a,*], Jari J.H. de Kroon [a], Michał Włodarczyk [b]

[a] *Eindhoven University of Technology, the Netherlands*
[b] *University of Warsaw, Poland*

A R T I C L E I N F O

Article history:
Received 8 January 2024
Received in revised form 4 October 2024
Accepted 4 October 2024
Available online 21 October 2024

Keywords:
Fixed-parameter tractability
Important separators
Secluded subgraphs

A B S T R A C T

The celebrated notion of important separators bounds the number of small (S, T)-separators in a graph which are 'farthest from S' in a technical sense. In this paper, we introduce a generalization of this powerful algorithmic primitive, tailored to undirected graphs, that is phrased in terms of k-*secluded* vertex sets: sets with an open neighborhood of size at most k. In this terminology, the bound on important separators says that there are at most 4^k maximal k-secluded connected vertex sets C containing S but disjoint from T. We generalize this statement significantly: even when we demand that $G[C]$ avoids a finite set $\mathcal{F}$ of forbidden induced subgraphs, the number of such maximal subgraphs is $2^{\mathcal{O}(k)}$ and they can be enumerated efficiently. This enumeration algorithm allows us to give improved parameterized algorithms for Connected k-Secluded $\mathcal{F}$-Free Subgraph and for deleting into scattered graph classes.

1. Introduction

Graph separations have played a central role in algorithmics since the discovery of min-cut/max-flow duality and the polynomial-time algorithm to compute a maximum flow [1]. Nowadays, more complex separation properties are crucial in the study of parameterized complexity, where the goal is to design algorithms for NP-hard problems whose running time can be bounded as $f(k) \cdot n^{\mathcal{O}(1)}$ for some function f that depends only on the *parameter* k of the input. There are numerous graph problems which either explicitly involve finding separations of a certain kind (such as Multiway Cut [2], Multicut [3,4], k-Way Cut [5], and Minimum Bisection [6]) or in which separation techniques turn out to be instrumental for an efficient solution (such as Directed Feedback Vertex Set [7] and Almost 2-SAT [8]).

The field of parameterized complexity has developed a robust toolbox of techniques based on graph separators, e.g., treewidth reduction [9], important separators [10], shadow removal [4], discrete relaxations [11–14], protrusion replacement [15], randomized contractions and recursive understanding [16–18], and flow augmentation [19,20]. These powerful techniques allowed a large variety of graph separation problems to be classified as fixed-parameter tractable. However, this power comes at a cost. The running times for many applications of these techniques are superexponential: of the form $2^{p(k)} \cdot n^{\mathcal{O}(1)}$ for a high-degree polynomial p, double-exponential, or even worse. Discrete relaxations form a notable exception, which we discuss in Section 5.

☆ An extended abstract of this work appeared in the Proceedings of the 34th International Symposium on Algorithms and Computation (ISAAC 2023).
* Corresponding author.
E-mail addresses: b.m.p.jansen@tue.nl (B.M.P. Jansen), j.j.h.d.kroon@tue.nl (J.J.H. de Kroon), m.wlodarczyk@mimuw.edu.pl (M. Włodarczyk).

https://doi.org/10.1016/j.jcss.2024.103597

The new algorithmic primitive we develop can be seen as an extension of important separators [10] [21, §8]. The study of important separators was pioneered by Marx [2,10] and refined by follow-up work by several authors [22,23], which was recognized by the EATCS-IPEC Nerode Prize 2020 [24]. The technique is used to bound the number of extremal (S,T)-separators in an n-vertex graph G with vertex sets S and T. The main idea is that, even though the number of distinct inclusion-minimal (S,T)-separators (which are vertex sets potentially intersecting $S \cup T$) of size at most k can be as large as $n^{\Omega(k)}$, the number of *important* separators which leave a maximal vertex set reachable from S, is bounded by 4^k. For Multiway Cut, a pushing lemma [2, Lem. 6] shows that there is always an optimal solution that contains an important separator, which leads to an algorithm solving the problem in time $2^{\mathcal{O}(k)} \cdot n^{\mathcal{O}(1)}$. Important separators also form a key ingredient for solving many other problems such as Multicut [3,4] and Directed Feedback Vertex Set [7].

For our purposes, it will be convenient to view the bound on the number of important separators through the lens of *secluded subgraphs*.

Definition 1. A vertex set $S \subseteq V(G)$ or induced subgraph $G[S]$ of an undirected graph G is said to be *k-secluded* if $|N_G(S)| \leq k$, that is, the number of vertices outside S which are adjacent to a vertex of S is bounded by k.

A vertex set S in a graph G is called *seclusion-maximal* with respect to a certain property Π if S satisfies Π and for all sets $S' \supsetneq S$ that satisfy Π we have $|N_G(S')| > |N_G(S)|$.

Hence a seclusion-maximal set with property Π is inclusion-maximal among all subsets with the same size neighborhood. Consequently, the number of inclusion-maximal k-secluded sets satisfying Π is at most the number of seclusion-maximal k-secluded sets with that property.

Using the terminology of seclusion-maximal subgraphs, the bound on the number of important (S,T)-separators of size at most k in a graph G is equivalent to the following statement: in the graph G' obtained from G by inserting a new source r adjacent to S, the number of *seclusion-maximal* k-secluded connected subgraphs C containing r but no vertex of T is bounded by 4^k. The neighborhoods of such subgraphs C correspond exactly to the important (S,T)-separators in G.

While a number of previously studied cut problems [9,25] place further restrictions on the vertex set that forms the separator (for example, requiring it to induce a connected graph or independent set) our generalization instead targets the structure of the k-secluded connected subgraph C. We will show that, for any fixed finite family $\mathcal{F}$ of graphs, the number of k-secluded connected subgraphs C as above which are seclusion-maximal with respect to satisfying the additional requirement that $G[C]$ contains no induced subgraph isomorphic to a member of $\mathcal{F}$ is still bounded by $2^{\mathcal{O}(k)}$. Observe that the case $\mathcal{F} = \emptyset$ corresponds to the original setting of important separators. Note that a priori, it is not even clear that the number of seclusion-maximal graphs of this form can be bounded by any function $f(k)$, let alone a single-exponential one.

Our contribution Having introduced the background of secluded subgraphs, we continue by stating our results exactly. This will be followed by a discussion on its applications.

For a finite set $\mathcal{F}$ of graphs we define $||\mathcal{F}|| := \max_{F \in \mathcal{F}} |V(F)|$, the maximum order of any graph in $\mathcal{F}$. We say that a graph is $\mathcal{F}$-free if it does not contain an *induced* subgraph isomorphic to a graph in $\mathcal{F}$. Our generalization of important separators is captured by the following theorem, in which we use $\mathcal{O}_{\mathcal{F}}(\ldots)$ to indicate that the hidden constant depends on $\mathcal{F}$.

Theorem 2. *Let $\mathcal{F}$ be a finite set of graphs. For any n-vertex graph G, non-empty vertex set $S \subseteq V(G)$, potentially empty $T \subseteq V(G) \setminus S$, and integer k, the number of k-secluded induced subgraphs $G[C]$ which are seclusion-maximal with respect to being connected, $\mathcal{F}$-free, and satisfying $S \subseteq C \subseteq V(G) \setminus T$, is bounded by $2^{\mathcal{O}_{\mathcal{F}}(k)}$. A superset of size $2^{\mathcal{O}_{\mathcal{F}}(k)}$ of these subgraphs can be enumerated in time $2^{\mathcal{O}_{\mathcal{F}}(k)} \cdot n^{||\mathcal{F}||+\mathcal{O}(1)}$ and polynomial space.*

The single-exponential bound given by the theorem is best-possible in several ways. Existing lower bounds on the number of important separators [21, Fig. 8.5] imply that even when $\mathcal{F} = \emptyset$ the bound cannot be improved to $2^{o(k)}$. The term $n^{||\mathcal{F}||}$ in the running time is unlikely to be avoidable, since even testing whether a single graph is $\mathcal{F}$-free is equivalent to Induced Subgraph Isomorphism and cannot be done in time $n^{o(||\mathcal{F}||)}$ [21, Thm. 14.21] assuming the Exponential Time Hypothesis (ETH) due to lower bounds for k-Clique.

The polynomial space bound applies to the internal space usage of the algorithm, as the output size may be exponential in k. More precisely, we consider polynomial-space algorithms equipped with a command that outputs an element and we require that for each element in the enumerated set, this command is called at least once. Theorem 2 guarantees that for each seclusion-maximal k-secluded induced subgraph $G[C]$ that is connected, $\mathcal{F}$-free, and satisfies $S \subseteq C \subseteq V(G) \setminus T$, the algorithm calls the output command at least once for C. The fact that the algorithm outputs a *superset* of the relevant subgraphs means that there could also be subgraphs $G[C']$ in the output for which some of these properties fail. One could also enumerate just the set in question (rather than its superset) by postprocessing the output and comparing each pair of enumerated subgraphs. However, storing the entire output requires exponential space.

By executing the enumeration algorithm for every singleton set S of the form $\{v\}$, $v \in V(G)$, and $T = \emptyset$, we immediately obtain the following.

Corollary 3. *Let $\mathcal{F}$ be a finite set of graphs. For any n-vertex graph G and integer k, the number of k-secluded induced subgraphs G[C] which are seclusion-maximal with respect to being connected and $\mathcal{F}$-free is $2^{\mathcal{O}_{\mathcal{F}}(k)} \cdot n$. A superset of size $2^{\mathcal{O}_{\mathcal{F}}(k)} \cdot n$ of these subgraphs can be enumerated in time $2^{\mathcal{O}_{\mathcal{F}}(k)} \cdot n^{||\mathcal{F}||+\mathcal{O}(1)}$ and polynomial space.*

Note that we require that the set $\mathcal{F}$ of forbidden induced subgraphs is finite. This is necessary in order to obtain a bound independent of n in Theorem 2. For example, the number of seclusion-maximal $(k=1)$-secluded connected subgraphs C containing a prescribed vertex r for which C induces an acyclic graph is already as large as $n-1$ in a graph consisting of a single cycle, since each way of omitting a vertex other than r gives such a subgraph. For this case, the forbidden induced subgraph characterization $\mathcal{F}$ consists of all cycles. Extending this example to a flower structure of k cycles of length n/k pairwise intersecting only in r shows that the number of seclusion-maximal k-secluded $\mathcal{F}$-free connected subgraphs containing r is $\Omega(n^k/k^k)$ and cannot be bounded by $f(k) \cdot n^{\mathcal{O}(1)}$ for any function f.

We give two applications of Theorem 2 to improve the running time of existing super-exponential (or even triple-exponential) parameterized algorithms to single-exponential, which is optimal under ETH. For each application, we start by presenting some context.

Application I: optimization over connected k-secluded $\mathcal{F}$-free subgraphs The computation of secluded versions of graph-theoretic objects such as paths [26–28], trees [29], Steiner trees [30], or feedback vertex sets [31], has attracted significant attention over recent years. This task becomes hard already for detecting k-secluded disconnected sets satisfying very simple properties. In particular, detecting a k-secluded independent set of size s is W[1]-hard when parameterized by $k+s$ [31].

Golovach, Heggernes, Lima, and Montealegre [32] suggested then to focus on *connected* k-secluded subgraphs and studied the problem of finding one, which belongs to a graph class $\mathcal{H}$, of maximum total weight. They therefore studied the Connected k-Secluded $\mathcal{F}$-Free Subgraph problem for a finite family $\mathcal{F}$ of forbidden induced subgraphs. Given an undirected graph G in which each vertex v has a positive integer weight $w(v)$, and an integer k, the problem is to find a maximum-weight connected k-secluded vertex set C for which $G[C]$ is $\mathcal{F}$-free. They presented an algorithm based on recursive understanding to solve the problem in time $2^{2^{2^{\mathcal{O}_{\mathcal{F}}(k \log k)}}} \cdot n^{\mathcal{O}_{\mathcal{F}}(1)}$. We improve the dependency on k to single-exponential.

Corollary 4. *For each fixed finite family $\mathcal{F}$,* Connected k-Secluded $\mathcal{F}$-Free Subgraph *can be solved in time $2^{\mathcal{O}_{\mathcal{F}}(k)} \cdot n^{||\mathcal{F}||+\mathcal{O}(1)}$ and polynomial space.*

This result follows directly from Corollary 3 since a maximum-weight k-secluded $\mathcal{F}$-free subgraph must be seclusion-maximal. Hence it suffices to check for each enumerated subgraph whether it is $\mathcal{F}$-free, and remember the heaviest one for which this is the case.

The parameter dependence of our algorithm for Connected k-Secluded $\mathcal{F}$-Free Subgraph is optimal under ETH. This follows from an easy reduction from Maximum Independent Set, which cannot be solved in time $2^{o(n)}$ under ETH [21, Thm. 14.6]. Finding a maximum independent set in an n-vertex graph G is equivalent to finding a maximum-weight triangle-free connected induced $(k=n)$-secluded subgraph in the graph G' that is obtained from G by inserting a universal vertex of weight n and setting the weights of all other vertices to 1. Consequently, an algorithm with running time $2^{o(k)} \cdot n^{\mathcal{O}(1)}$ for Connected k-Secluded Triangle-Free Induced Subgraph would violate ETH and our parameter dependence is already optimal for $\mathcal{F} = \{K_3\}$.

Application II: deletion to scattered graph classes When there are several distinct graph classes (e.g., split graphs and claw-free graphs) on which a problem of interest (e.g. Vertex Cover) becomes tractable, it becomes relevant to compute a minimum vertex set whose removal ensures that each resulting component belongs to one such tractable class. This can lead to fixed-parameter tractable algorithms for solving the original problem on inputs which are *close* to such so-called *islands of tractability* [33]. The corresponding optimization problem has been coined the deletion problem to *scattered* graph classes [34,35]. Jacob, Majumdar, and Raman [36] (later joined by de Kroon for the journal version [34]) considered the $(\Pi_1, \dots, \Pi_d)$-Deletion problem; given hereditary graph classes $\Pi_1, \dots, \Pi_d$, find a set $X \subseteq V(G)$ of at most k vertices such that each connected component of $G-X$ belongs to Π_i for some $i \in [d]$. Here d is seen as a constant. When the set of forbidden induced subgraphs $\mathcal{F}_i$ of Π_i is finite for each $i \in [d]$, they showed [34, Lem. 12] that the problem is solvable in time $2^{q(k)} \cdot n^{\mathcal{O}_{\Pi}(1)}$, where $q(k) = 4k^{10(pd)^2+4}+1$. Here p is the maximum number of vertices of any forbidden induced subgraph.

Using Theorem 2 as a black box, we obtain a single-exponential algorithm for this problem.

Theorem 5. $(\Pi_1, \dots, \Pi_d)$-Deletion *can be solved in time $2^{\mathcal{O}_{\Pi}(k)} \cdot n^{\mathcal{O}_{\Pi}(1)}$ and polynomial space when each graph class Π_i is characterized by a finite set $\mathcal{F}_i$ of (not necessarily connected) forbidden induced subgraphs.*

The main idea behind the algorithm is the following. For an arbitrary vertex v, either it belongs to the solution (which we can test by removing v and decreasing k by one), or we may assume that in the graph that results by removing the solution, the vertex v belongs to a connected component that forms a *seclusion-maximal* connected k-secluded $\mathcal{F}_i$-free

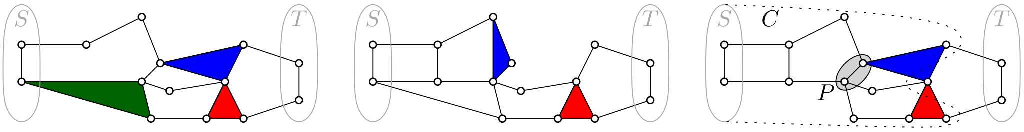

Fig. 1. Illustration of the branching steps for enumerating triangle-free k-secluded subgraphs for $k = 3$. Left: the green triangle intersects S; we branch to guess which vertex belongs to $N(C)$. Middle: setting where $2 = \lambda^L(S, T) < \lambda^L(S, T \cup V(\mathcal{U})) = 3$; adding the top triangle to T increases λ^L. The set $\mathcal{U}$ consists of the colored triangles. Right: setting where $\lambda^L(S, T) = \lambda^L(S, T \cup V(\mathcal{U})) = 2$, with a corresponding farthest separator P. In this case every seclusion-maximal triangle-free set $C \supseteq S$ must be a superset of the reachability set of S in $G - P$. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

induced subgraph C of G for some $i \in [d]$. By Corollary 4, the number of such subgraphs C is bounded by $2^{\mathcal{O}_\Pi(k)}$. For each choice of C, we can recurse on the graph $G - N_G[C]$ with a budget of $k' := k - |N_G(C)|$ to determine whether there is a solution in which C remains as a connected component. By exploiting the fact that in most recursive calls the parameter decreases by more than a constant (cf. [21, Thm. 8.19]), this yields the desired running time. Prior to our work, single-exponential algorithms were only known for a handful of ad-hoc cases where $d = 2$, such as deleting to a graph in which each component is a tree or a clique [35], or when one of the sets of forbidden induced subgraphs $\mathcal{F}_i$ contains a path.

Similarly as our first application, the resulting algorithm for $(\Pi_1, \ldots, \Pi_d)$-Deletion is ETH-tight: the problem is a strict generalization of k-Vertex Cover, which is known not to admit an algorithm with running time $2^{o(k)} \cdot n^{\mathcal{O}(1)}$ unless ETH fails.

Techniques The proof of Theorem 2 is based on a bounded-depth search tree algorithm with a nontrivial progress measure. By adding vertices to S or T in branching steps of the enumeration algorithm, the sets grow and the size of a minimum (S, T)-separator increases accordingly. We will be interested in separators that are disjoint from S, which we call *left-restricted*. The size of a minimum left-restricted (S, T)-separator is an important progress measure for the algorithm: if it ever exceeds k, there can be no k-secluded set containing all of S and none of T and therefore the enumeration is finished.

The branching steps are informed by the farthest minimum left-restricted (S, T)-separator (see Lemma 9), similarly as the enumeration algorithm for important separators, but are significantly more involved because we have to handle the forbidden induced subgraphs. A distinctive feature of our algorithm is that the decision made by branching can be to add certain vertices to the set T, while the important-separator enumeration only branches by enriching S. A key step is to use submodularity to infer that a certain vertex set is contained in *all* seclusion-maximal secluded subgraphs under consideration when other branching steps are inapplicable.

As an illustrative example consider the case $\mathcal{F} = \{K_3\}$, that is, we want to enumerate seclusion-maximal vertex sets $C \subseteq V(G) \setminus T$, $C \supseteq S$, which induce connected triangle-free subgraphs with at most k neighbors. Let $\lambda^L(S, T)$ denote the size of a minimum left-restricted (S, T)-separator. Then $\lambda^L(S, T)$ corresponds to the minimum possible size of $N(C)$. Similarly to the enumeration algorithm for important separators, we keep track of two measures: (M1) the value of k, and (M2) the gap between k and $\lambda^L(S, T)$. We combine them into a single progress measure which is bounded by $2k$ and decreases during branching.

The first branching scenario occurs when there is some triangle in the graph G which intersects or is adjacent to S; then we guess which of its vertices should belong to $N(C)$, remove it from the graph, and decrease k by one. Otherwise, let $\mathcal{U} = \{U_1, \ldots, U_d\}$ be the collection of all vertex sets of triangles in G (which are now disjoint from S). When there exists a triangle U_i whose addition to T increases the value $\lambda^L(S, T)$, we branch into two possibilities: either U_i is disjoint from $N[C]$—then we set $T \leftarrow T \cup U_i$ so the measure (M2) decreases—or U_i intersects $N(C)$—then we perform branching as above. In the remaining case that adding any single triangle U_i to T leaves $\lambda^L(S, T)$ unchanged, we show that all the triangles are separated from S by the minimum left-restricted (S, T)-separator closest to S; hence the value of $\lambda^L(S, T)$ equals the value of $\lambda^L(S, T \cup V(\mathcal{U}))$. Next, let P be the farthest minimum left-restricted $(S, T \cup V(\mathcal{U}))$-separator; we use submodularity to justify that we can now safely add to S all the vertices reachable from S in $G - P$. This allows us to assume that when $u \in P$ then either $u \in N(C)$ or $u \in C$, which leads to the last branching strategy. We either delete u (so k drops) or add u to S; note that in this case the progress measure may not change directly. The key observation is that adding u to S invalidates the farthest $(S, T \cup V(\mathcal{U}))$-separator P and now we are promised to make progress in the very next branching step. The different branching scenarios are illustrated in Fig. 1. When there is nothing left to branch on, we have found a seclusion-maximal triangle-free subgraph and we give it as output. This concludes the algorithm for $\mathcal{F} = \{K_3\}$.

The only property of K_3 that we have relied on is connectivity: if a triangle intersects a triangle-free set C then it must intersect $N(C)$ as well. This is no longer true when $\mathcal{F}$ contains a disconnected graph. For example, the forbidden family for the class of split graphs includes $2K_2$. A subgraph of $F \in \mathcal{F}$ that can be obtained by removing some components from F is called a *partial forbidden graph*. We introduce a third measure to keep track of how many different partial forbidden graphs appear as induced subgraph in $G[S]$. The main difficulty in generalizing the previous approach lies in justification of the greedy argument: when P is a farthest minimum separator between S and a certain set then we want to replace S with the set S' of vertices reachable from S in $G - P$. In the setting of connected obstacles this fact could be proven easily because S' was disjoint from all the obstacles. The problem is now it may contain some partial forbidden subgraphs. We handle this issue by defining P in such a way that the sets of partial forbidden graphs appearing in $G[S]$ and $G[S']$ are the same and giving a rearrangement argument about subgraph isomorphisms. This allows us to extend the analysis to any family $\mathcal{F}$ of forbidden subgraphs.

Organization The remainder of the paper is organized as follows. We provide formal preliminaries in Section 2. The algorithm for enumerating secluded $\mathcal{F}$-free subgraphs is presented in Section 3. Then in Section 4 we apply it to improve the running times of the two discussed problems. We conclude in Section 5.

2. Preliminaries

2.1. Graphs and separators

We consider finite, simple, undirected graphs. We denote the vertex and edge sets of a graph G by $V(G)$ and $E(G)$ respectively, with $|V(G)| = n$ and $|E(G)| = m$. For a set of vertices $S \subseteq V(G)$, by $G[S]$ we denote the graph induced by S. We use shorthand $G - v$ and $G - S$ for $G[V(G) \setminus \{v\}]$ and $G[V(G) \setminus S]$, respectively. The *open neighborhood* $N_G(v)$ of $v \in V(G)$ is defined as $\{u \in V(G) \mid \{u, v\} \in E(G)\}$. The *closed neighborhood* of v is $N_G[v] = N_G(v) \cup \{v\}$. For $S \subseteq V(G)$, we have $N_G[S] = \bigcup_{v \in S} N_G[v]$ and $N_G(S) = N_G[S] \setminus S$. The set C is called *connected* if the graph $G[C]$ is connected. For a graph G and vertex set S, the operation of *identifying* vertex set S into a single vertex s^* results in the graph G' obtained from $G - S$ by inserting a new vertex s^* that is made adjacent to all vertices of $N_G(S)$.

We proceed by introducing notions concerning separators which are crucial for the branching steps of our algorithms. For two sets $S, T \subseteq V(G)$ in a graph G, a set $P \subseteq V(G)$ is an *unrestricted* (S, T)*-separator* if no connected component of $G - P$ contains a vertex from both $S \setminus P$ and $T \setminus P$. Note that such a separator may intersect $S \cup T$. Equivalently, P is an (S, T)-separator if each (S, T)-path contains a vertex of P. A *restricted* (S, T)*-separator* is an unrestricted (S, T)-separator P which satisfies $P \cap (S \cup T) = \emptyset$. A *left-restricted* (S, T)*-separator* is an unrestricted (S, T)-separator P which satisfies $P \cap S = \emptyset$. Let $\lambda^{\mathrm{L}}_G(S, T)$ denote the minimum size of a left-restricted (S, T)-separator, or $+\infty$ if no such separator exists (which happens when $S \cap T \neq \emptyset$).

Theorem 6 *(Ford-Fulkerson). There is an algorithm that, given an n-vertex m-edge graph $G = (V, E)$, disjoint sets $S, T \subseteq V(G)$, and an integer k, runs in time $\mathcal{O}(k(n + m))$ and determines whether there exists a restricted (S, T)-separator of size at most k. If so, then the algorithm returns a separator of minimum size.*

By the following observation we can translate properties of restricted separators into properties of left-restricted separators.

Observation 7. *Let G be a graph and $S, T \subseteq V(G)$. Consider the graph G' obtained from G by adding a new vertex t adjacent to each $v \in T$. Then $P \subseteq V(G)$ is a left-restricted (S, T)-separator in G if and only if P is a restricted (S, t)-separator in G'.*

2.2. Extremal separators and submodularity

The following submodularity property of the cardinality of the open neighborhood is well-known; cf. [37, §44.12] and [38, Fn. 3].

Lemma 8 *(Submodularity). Let G be a graph and $A, B \subseteq V(G)$. Then the following holds:*

$$|N_G(A)| + |N_G(B)| \geq |N_G(A \cap B)| + |N_G(A \cup B)|.$$

For a graph G and vertex sets $S, P \subseteq V(G)$, we denote by $R_G(S, P)$ the set of vertices which can be reached in $G - P$ from at least one vertex in the set $S \setminus P$. When P is an (S, T)-separator, then $R_G(S, P)$ contains no vertex of T by definition. If P is an (S, T)-separator that is *inclusion-minimal*, meaning no proper subset is an (S, T)-separator, we have $P = N_G(R_G(S, P))$: all vertices of $N_G(R_G(S, P))$ must be contained in P (a neighbor of a reachable vertex must itself be reachable, unless it is deleted by the separator), while the set $N_G(R_G(S, P))$ suffices to intersect all (S, T)-paths since $R_G(S, P)$ contains no vertex of T. We refer to Proposition 8.48 in the textbook by Cygan et al. [21] for further background. Since any *minimum* separator is inclusion-minimal, we have $P = N_G(R_G(S, P))$ for all minimum (left-restricted) (S, T)-separators. We will frequently use this property in our proofs.

The following lemma applies known properties of extremal restricted separators in the context of left-restricted separators. See Fig. 2 for an illustration. The first item proves that there are unique choices for a closest and farthest (from S) minimum left-restricted separator. For any other minimum left-restricted separator, the set of vertices that remains reachable from S after removal of the separator is a superset of what remains for the closest separator, and a subset of what remains for the farthest separator. These separators can be computed efficiently using (network) flow techniques. The last item of the lemma uses these separators to determine when adding a vertex to S or T increases $\lambda^{\mathrm{L}}_G(S, T)$, which can be characterized in terms of properties of the closest and farthest separators. Note that the characterization is not symmetric because we work with the non-symmetric notion of a left-restricted separator.

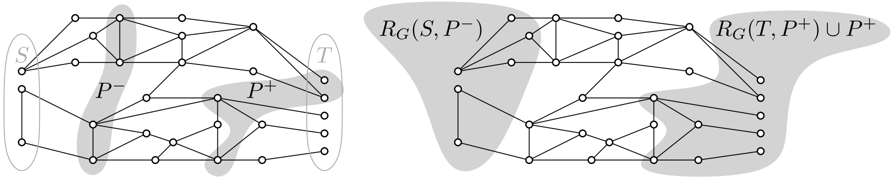

Fig. 2. Left: Illustration of the closest (P^-) and farthest (P^+) minimum left-restricted (S, T)-separators. Note that the minimum size of an *unrestricted* (S, T)-separator in this graph is three, since in the unrestricted case we may choose all of S into the separator. Right: Illustration of the vertex sets treated in Lemma 9(3). Adding a vertex v to the set S (from the left figure) increases the size of a minimum left-restricted separator if and only if $v \in R_G(T, P^+) \cup P^+$. Similarly, adding a vertex v to the set T increases the size of a minimum left-restricted separator if and only if $v \in R_G(S, P^-)$.

Lemma 9. *Let G be a graph and $S, T \subseteq V(G)$ be two disjoint vertex sets.*

1. *There exist unique minimum left-restricted (S, T)-separators P^- and P^+, such that for each minimum left-restricted (S, T)-separator P, we have:*

$$R_G(S, P^-) \subseteq R_G(S, P) \subseteq R_G(S, P^+).$$

   We refer to P^- and P^+ as the closest *and* farthest *minimum left-restricted (S, T)-separator, respectively.*
2. *If a minimum left-restricted (S, T)-separator has size k, then P^- and P^+ can be identified in $\mathcal{O}(k(n+m))$ time.*
3. *For any vertex $v \in V(G)$, the following holds:*
   (a) *$\lambda^{L}_{G}(S \cup \{v\}, T) > \lambda^{L}_{G}(S, T)$ if and only if $v \in R_G(T, P^+) \cup P^+$.*
   (b) *$\lambda^{L}_{G}(S, T \cup \{v\}) > \lambda^{L}_{G}(S, T)$ if and only if $v \in R_G(S, P^-)$.*

Proof. Before proving the three items, we introduce some concepts and an auxiliary claim. Let $k = \lambda^{L}_{G}(S, T)$ denote the cardinality of a minimum left-restricted (S, T)-separator. Note that such a separator exists since S and T are assumed to be disjoint. We call a set L with $S \subseteq L \subseteq V(G) \setminus T$ a *left set*. Note that for any left set L, the set $N_G(L)$ is a left-restricted (S, T)-separator: it is disjoint from S since $S \subseteq L$, and separates since all paths from S to T must leave the set L via a vertex of $N_G(L)$. Each *minimum* left-restricted (S, T)-separator P can be written as the neighborhood of a left set L by taking $L = R_G(S, P)$.

Claim 1. *If L_1, L_2 are left sets such that both $N_G(L_1)$ and $N_G(L_2)$ are minimum left-restricted (S, T)-separators of size k, then the sets $L_1 \cup L_2$ and $L_1 \cap L_2$ are left sets. Furthermore, $N_G(L_1 \cap L_2)$ and $N_G(L_1 \cup L_2)$ are minimum left-restricted (S, T)-separators.*

Proof. Since L_1 and L_2 both contain all of S and none of T, the same is true for their union and intersection. Hence $L_1 \cup L_2$ and $L_1 \cap L_2$ are left sets. Let us consider the size of the open neighborhood of their union and intersection, by applying the submodular inequality of Lemma 8. We obtain:

$$|N_G(L_1)| + |N_G(L_2)| \geq |N_G(L_1 \cap L_2)| + |N_G(L_1 \cup L_2)|.$$

By assumption, the left-hand side is equal to $2k$. Since both $L_1 \cap L_2$ and $L_1 \cup L_2$ are left sets, their neighborhoods form left-restricted (S, T)-separators. Since the minimum size of a left-restricted (S, T)-separator is k by assumption, both $|N_G(L_1 \cap L_2)|$ and $|N_G(L_1 \cup L_2)|$ are at least k. Hence the right-hand side is at least $2k$. Since the left-hand side is at least the right-hand side, we must have equality. This implies $|N_G(L_1 \cap L_2)| = |N_G(L_1 \cup L_2)| = k$ and proves the claim. ⌟

Existence of closest and farthest separators We now prove Item 1 using the previous claim. Let $\mathcal{L} \subseteq 2^{V(G)}$ denote the family of all left sets in G whose open neighborhood has size k, and whose open neighborhood is therefore a minimum restricted (S, T)-separator. Define $L^- := \bigcap_{L \in \mathcal{L}} L$ and $L^+ := \bigcup_{L \in \mathcal{L}} L$. Since all left sets in $\mathcal{L}$ have an open neighborhood of size k, Claim 1 shows that every time we take the intersection (or union) of two left sets in $\mathcal{L}$, the resulting set is another left set with an open neighborhood of size k. This shows that $P^- := N_G(L^-)$ and $P^+ := N_G(L^+)$ have size k, and that they are left-restricted (S, T)-separators since L^- and L^+ are left sets. We claim that P^- and P^+ satisfy the conditions for being the closest and farthest left-restricted minimum (S, T)-separators.

Consider an arbitrary minimum restricted (S, T)-separator P. We prove that $R_G(S, P^-) \subseteq R_G(S, P) \subseteq R_G(S, P^+)$. Since P is a minimum restricted (S, T)-separator, it must be inclusion-minimal, so that $P = N_G(R_G(S, P))$. Note that $R_G(S, P)$ is a left set: it contains all vertices of S (since the separator is disjoint from S), and no vertex of T (since no vertex of S can reach T in $G - P$). Since P is minimum, the size of $N_G(R_G(S, P))$ is k. So the left set $R_G(S, P)$ is contained in $\mathcal{L}$. Hence $R_G(S, P)$ is one of the sets whose intersection is taken when computing L^-, and one of the sets whose union is taken when computing L^+. It follows that $L^- \subseteq R_G(S, P) \subseteq L^+$. Since L^- is a left set, it contains all vertices of S. Hence the only vertices of G that can be reachable from S in $G - N_G(L^-) = G - P^-$, are those of L^-. It follows that $R_G(S, P^-) \subseteq L^- \subseteq R_G(S, P)$. It remains to argue that $R_G(S, P) \subseteq R_G(S, P^+)$. Since $R_G(S, P) \subseteq L^+$, any path Q in the graph $G - P$ from S to a vertex $v \in R_G(S, P)$ is contained entirely within L^+ and is therefore disjoint from $N_G(L^+) = P^+$. Hence any such Q is also a path from S to v in $G - P^+$, which implies $v \in R_G(S, P^+)$. This proves $R_G(S, P) \subseteq R_G(S, P^+)$.

All that remains to establish Item 1 is the uniqueness of P^- and P^+, which follows easily from their properties. We give the uniqueness argument for P^-; the one for P^+ is symmetric. Suppose for a contradiction that there are two distinct minimum left-restricted (S, T)-separators P_1^-, P_2^- that satisfy the requirements of a closest separator: for any minimum left-restricted separator P we have $R_G(S, P_1^-) \subseteq R_G(S, P)$ and $R_G(S, P_2^-) \subseteq R_G(S, P)$. By instantiating the first statement for $P = P_2^-$ we have $R_G(S, P_1^-) \subseteq R_G(S, P_2^-)$, and by instantiating the second for $P = P_1^-$ we get the converse $R_G(S, P_2^-) \subseteq R_G(S, P_1^-)$. Hence $R_G(S, P_1^-) = R_G(S, P_2^-)$. Since P_1^- and P_2^- are inclusion-minimal (S, T)-separators, we have $P_1^- = N_G(R_G(S, P_1^-))$ and $P_2^- = N_G(R_G(S, P_2^-))$. But then $P_1^- = P_2^-$, a contradiction to the assumption they are distinct.

Efficiently computing closest and farthest separators While the definition of P^- and P^+ involved an intersection or union of (potentially exponentially) many sets, it is known that they can also be computed efficiently using flow techniques. See [21, Thm. 8.5] for the edge-based variant of this statement, or [38, §3.2] for the same concept with slightly different terminology. We sketch the main ideas for completeness.

For the purpose of computing P^+ and P^-, we may identify S into a single vertex s^+: this is harmless because a left-restricted separator is disjoint from S. Then we may insert a new vertex t^- into the graph, which becomes adjacent to all vertices of T. Let G' denote the resulting graph. Then a left-restricted (S, T)-separator in G corresponds to a restricted (s^+, t^-)-separator in G'. We transform G' into an edge-capacitated directed flow network D in which s^+ is the source and t^- is the sink. All remaining vertices $v \in V(G') \setminus \{s^+, t^-\}$ are split into two representatives v^-, v^+ connected by an arc (v^-, v^+) of capacity 1. For each edge $uv \in E(G')$ with $u, v \in V(G') \setminus \{s^+, t^-\}$ we add arcs $(u^+, v^-), (v^+, u^-)$ of capacity 2. For edges of the form s^+v we add an arc (s^+, v^-) of capacity 2 to D. Similarly, for edges of the form t^-v we add an arc (v^+, t^-) of capacity 2. Then the minimum size k of a restricted (s^+, t^-)-separator in G' equals the maximum flow value in the constructed network, which can be computed by k rounds of the Ford-Fulkerson algorithm. Each round can be implemented to run in time $\mathcal{O}(n + m)$. From the state of the residual network when Ford-Fulkerson terminates we can extract L^- and L^+ as follows: the set L^- contains all vertices $v \in V(G')$ for which the source can reach v^- but not v^+ in the final residual network. Similarly, L^+ contains all vertices $v \in V(G')$ for which v^+ can reach the sink but v^- cannot. It then suffices to take $P^+ := N_{G'}(L^+)$ and $P^- := N_{G'}(L^-)$. This establishes Item 3.

Vertices that increase λ^{L} Using the notions of left sets developed above, together with the definitions of L^- and L^+ from the proof of Item 1, we can now prove Item 3 which characterizes when adding a vertex v to S or T increases λ^{L}. In both cases it is clear that the left-hand side is at least as large as the right-hand side. Hence it suffices to argue that the inequality is strict precisely when v belongs to the stated set.

We first show Item 3a, that $\lambda^{\mathsf{L}}_G(S \cup \{v\}, T) > \lambda^{\mathsf{L}}_G(S, T)$ if and only if $v \in R_G(T, P^+) \cup P^+$. So consider an arbitrary vertex $v \in V(G)$. If $v \notin R_G(T, P^+) \cup P^+$, then P^+ is also a left-restricted $(S \cup \{v\}, T)$-separator: it is a separator since v is not reachable from T in $G - P^+$, and it is left-restricted since $v \notin P^+$. So in this case, $\lambda^{\mathsf{L}}_G(S \cup \{v\}, T) = \lambda^{\mathsf{L}}_G(S, T)$. Hence if the right-hand condition does not hold, then the left-hand condition does not hold either.

Now suppose that the left-hand condition fails, so that $\lambda^{\mathsf{L}}_G(S \cup \{v\}, T) \leq \lambda^{\mathsf{L}}_G(S, T)$, which can only happen if the two quantities are equal. We argue that the right-hand condition fails as well. Let P be a minimum left-restricted $(S \cup \{v\}, T)$-separator in G, which has size $\lambda^{\mathsf{L}}_G(S, T) = k$ by assumption. Let $L \subseteq V(G - P)$ be the set of vertices of $G - P$ that cannot reach any vertex of T in $G - P$. So then $v \in L$ and L contains all vertices of S but none of T, since $S \cap P = \emptyset$ and P is a separator. Hence L is a left set. All vertices of $N_G(L)$ belong to P (otherwise they would be reachable themselves), and $P = N_G(L)$ since P is a minimum left-restricted (S, T)-separator as its size is k. So vertex set L belongs to the family $\mathcal{L}$ defined in the proof of Item 1. This implies that $L \subseteq L^+$, for the set L^+ defined above. But then v is not in the open neighborhood of L^+, since $v \in L \subseteq L^+$. It follows that $v \notin P^+$. We also claim that $v \notin R_G(T, P^+)$: all paths from $v \in L^+$ to T go through $N_G(L^+) = P^+$, since $v \in L^+$ while T is disjoint from L^+. So $v \notin R_G(T, P^+)$. Summarizing, we argued that if $\lambda^{\mathsf{L}}_G(S \cup \{v\}, T) \leq \lambda^{\mathsf{L}}_G(S, T)$, then $v \notin R_G(T, P^+) \cup P^+$. This proves the equivalence of Item 3a.

It remains to prove Item 3b. The argument is similar, but slightly easier. If $v \notin R_G(S, P^-)$, then P^- is also an $(S, T \cup \{v\})$-separator so $\lambda^{\mathsf{L}}_G(S, T \cup \{v\}) \leq \lambda^{\mathsf{L}}(S, T)$. Hence if the right-hand side of Item 3b fails, the left-hand fails as well. We now prove the converse. Suppose the left-hand side fails, so that $\lambda^{\mathsf{L}}_G(S, T \cup \{v\}) \leq \lambda^{\mathsf{L}}_G(S, T)$, and let P be a minimum left-restricted $(S, T \cup \{v\})$-separator in G, which has size $\lambda^{\mathsf{L}}_G(S, T)$ by assumption and is therefore a minimum left-restricted (S, T)-separator. Since P separates S from $T \cup \{v\}$, we have $v \notin R_G(S, P)$. By Item 1 we have $R_G(S, P^-) \subseteq R_G(S, P)$. Since v is not contained in the latter set, it is not contained in the former either. Hence $v \notin R_G(S, P^-)$, which proves that the right-hand side fails as well. Hence the two statements are equivalent.

This completes the proof of Item 3 and finishes the proof of Lemma 9. □

The following lemma captures the idea that if $\lambda^{\mathsf{L}}_G(S, T \cup Z) > \lambda^{\mathsf{L}}_G(S, T)$, then there is a single vertex from Z whose addition to T already increases the size of a minimum left-restricted (S, T)-separator. We will use it to argue that when it is cheaper to separate S from T than to separate S from T together with all obstacles of a certain form, then there is already a single vertex from one such obstacle which causes this increase.

Lemma 10. *Let G be a graph, $S \subseteq V(G)$, and $T, Z \subseteq V(G) \setminus S$. If there is no vertex $v \in Z$ such that $\lambda^{\mathrm{L}}_G(S, T \cup \{v\}) > \lambda^{\mathrm{L}}_G(S, T)$, then $\lambda^{\mathrm{L}}_G(S, T) = \lambda^{\mathrm{L}}_G(S, T \cup Z)$. Furthermore if $\lambda^{\mathrm{L}}_G(S, T) \leq k$, then in $\mathcal{O}(k(n+m))$ time we can either find a vertex $v \in Z$ such that $\lambda^{\mathrm{L}}_G(S, T \cup \{v\}) > \lambda^{\mathrm{L}}_G(S, T)$ or determine that no such vertex exists.*

Proof. Let P^- be the minimum left-restricted (S, T)-separator which is closest to S. If for every $v \in Z$ the value of $\lambda^{\mathrm{L}}_G(S, T \cup \{v\})$ equals $\lambda^{\mathrm{L}}_G(S, T)$ then Lemma 9(3) implies that each $v \in Z$ lies outside $R_G(S, P^-)$ so $Z \cap R_G(S, P^-) = \emptyset$. Then P^- is a left-restricted $(S, T \cup Z)$-separator of size $\lambda^{\mathrm{L}}_G(S, T)$.

On the other hand, if there is a vertex $v \in Z$ for which $\lambda^{\mathrm{L}}_G(S, T \cup \{v\}) > \lambda^{\mathrm{L}}_G(S, T)$ then Lemma 9(3) implies that $v \in R_G(S, P^-)$. To detect such a vertex, we therefore compute the closest minimum left-restricted (S, T)-separator P^- in time $\mathcal{O}(k(n+m))$ via Lemma 9. Then we compute $R_G(S, P^-)$, which can trivially be done in time $\mathcal{O}(n+m)$. If it contains a vertex of Z, we output such a vertex; otherwise we report that there is no $v \in Z$ with $\lambda^{\mathrm{L}}_G(S, T \cup \{v\}) > \lambda^{\mathrm{L}}_G(S, T)$. □

Finally, the last lemma of this section uses submodularity to argue that the neighborhood size of a vertex set C with $S \subseteq C \subseteq V(G) \setminus T$ does not increase when taking its union with the reachable set $R_G(S, P)$ with respect to a minimum left-restricted (S, T)-separator P.

Lemma 11. *If $P \subseteq V(G)$ is a minimum left-restricted (S, T)-separator in a graph G and $S' = R_G(S, P)$, then for any set C with $S \subseteq C \subseteq V(G) \setminus T$ we have $|N_G(C \cup S')| \leq |N_G(C)|$.*

Proof. Observe that since P is a minimum left-restricted (S, T)-separator, we have $|P| = \lambda^{\mathrm{L}}_G(S, T)$ and $P = N_G(S')$. We apply the submodular inequality of Lemma 8 to the sets C and S' to obtain:

$$|N_G(C)| + |N_G(S')| \geq |N_G(C \cup S')| + |N_G(C \cap S')| \geq |N_G(C \cup S')| + \lambda^{\mathrm{L}}_G(S, T).$$

Here the last step comes from the fact that $S \subseteq S' \subseteq V(G) \setminus T$ since it is the set reachable from S with respect to a left-restricted (S, T)-separator, so that $C \cap S'$ contains all of S and is disjoint from T. This implies that $N_G(C \cap S')$ is a left-restricted (S, T)-separator, so that $|N_G(C \cap S')| \geq \lambda^{\mathrm{L}}_G(S, T)$.

As $|N_G(S')| = |P| = \lambda^{\mathrm{L}}_G(S, T)$, canceling these terms from both sides gives $|N_G(C)| \geq |N_G(C \cup S')|$ which completes the proof. □

3. The enumeration algorithm

We need the following concept to deal with forbidden subgraphs which may be disconnected.

Definition 12. For a fixed finite family $\mathcal{F}$ of forbidden induced subgraphs, a *partial forbidden graph of* $\mathcal{F}$ is a graph F' obtained from some $F \in \mathcal{F}$ by deleting zero or more connected components. (So each $F \in \mathcal{F}$ itself is also considered a partial forbidden graph.)

When the family $\mathcal{F}$ is clear from the context, we will just refer to such a graph F' as a *partial forbidden graph*.

We use the following notation to work with induced subgraph isomorphisms. An induced subgraph isomorphism from H to G is an injection $\phi\colon V(H) \to V(G)$ such that for all distinct $u, v \in V(H)$ we have $\{u, v\} \in E(H)$ if and only if $\{\phi(u), \phi(v)\} \in E(G)$. For a vertex set $U \subseteq V(H)$ we let $\phi(U) := \{\phi(u) \mid u \in U\}$. For a (not necessarily induced) subgraph H' of H we write $\phi(H')$ instead of $\phi(V(H'))$.

The following definition will be important to capture the progress of the recursive algorithm. See Fig. 3 for an illustration.

Definition 13. We say that a vertex set $U \subseteq V(G)$ *enriches* a vertex set $S \subseteq V(G)$ with respect to $\mathcal{F}$ if there exists a partial forbidden graph F' such that $G[S \cup U]$ contains an induced subgraph isomorphic to F' but $G[S]$ does not. We call such a set U an *enrichment*.

An enrichment U is called *tight* if $U = \phi(F') \setminus S$ for some induced subgraph isomorphism $\phi\colon V(F') \to V(G)$ from some partial forbidden graph F' for which $G[S]$ does not contain an induced subgraph isomorphic to F'.

This definition implies that $|U| \leq ||\mathcal{F}||$ for any tight enrichment U: any partial forbidden graph F' has at most $||\mathcal{F}||$ vertices, so that $U = \phi(F') \setminus S$ has cardinality at most $||\mathcal{F}||$.

The following observation will be used to argue for the correctness of the recursive scheme.

Observation 14. *Let G be a graph containing disjoint sets $S, T \subseteq V(G)$ and let $C \subseteq V(G)$ be seclusion-maximal with respect to being connected, $\mathcal{F}$-free, k-secluded and satisfying $S \subseteq C \subseteq V(G) \setminus T$. For each $v \in N_G(C)$ it holds that C is seclusion-maximal in $G - v$ with respect to being connected, $\mathcal{F}$-free, $(k-1)$-secluded and satisfying $S \subseteq C \subseteq V(G - v) \setminus T$.*

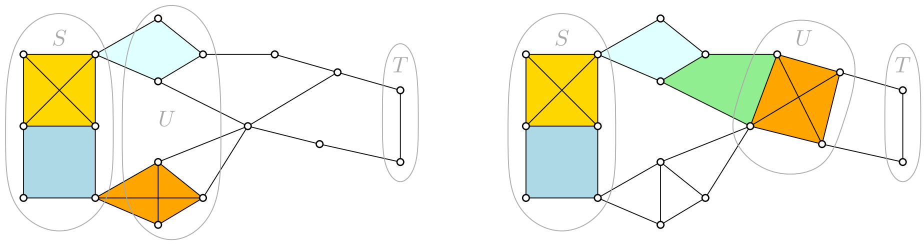

Fig. 3. Illustration of the idea of enrichment and the branching steps in the proof of Theorem 2. Here $F = C_4 \uplus K_4$. Left: The graph $G[S]$ contains C_4 and K_4, but not F. The set U enriches S since $G[S \cup U]$ contains a new partial forbidden graph F. Every component of $G[U]$ is adjacent to S, so Step 3 applies. Right: The two top copies of C_4 do not enrich S. One of them intersects the only copy of K_4 in $G[S]$; the other one is adjacent to the only copy of K_4, while F has to appear as an induced subgraph. However the connected set U enriches S and it gets detected in Step 4. In both cases the enrichments are tight.

Note that we get an implication only in one way (being seclusion-maximal in G implies being seclusion-maximal in $G - v$, not the other way around), which is the reason why we output a superset of the sought set in Theorem 2.

With these ingredients, we present the enumeration algorithm. We re-state the corresponding theorem for completeness. Recall that $||\mathcal{F}|| = \max_{F \in \mathcal{F}} |V(F)|$ denotes the maximum order of any graph in $\mathcal{F}$.

Theorem 2. *Let $\mathcal{F}$ be a finite set of graphs. For any n-vertex graph G, non-empty vertex set $S \subseteq V(G)$, potentially empty $T \subseteq V(G) \setminus S$, and integer k, the number of k-secluded induced subgraphs $G[C]$ which are seclusion-maximal with respect to being connected, $\mathcal{F}$-free, and satisfying $S \subseteq C \subseteq V(G) \setminus T$, is bounded by $2^{\mathcal{O}_{\mathcal{F}}(k)}$. A superset of size $2^{\mathcal{O}_{\mathcal{F}}(k)}$ of these subgraphs can be enumerated in time $2^{\mathcal{O}_{\mathcal{F}}(k)} \cdot n^{||\mathcal{F}||+\mathcal{O}(1)}$ and polynomial space.*

Proof. Algorithm $\mathsf{Enum}_{\mathcal{F}}(G, S, T, k)$ solves the enumeration task as follows.

1. Stop the algorithm if one of the following holds:
   (a) $\lambda^{\mathsf{L}}_G(S, T) > k$,
   (b) the vertices of S are not contained in a single connected component of G, or
   (c) the graph $G[S]$ contains an induced subgraph isomorphic to some $F \in \mathcal{F}$.
   There are no secluded subgraphs satisfying all imposed conditions.
2. If the connected component C of G which contains S is $\mathcal{F}$-free and includes no vertex of T: output C and stop.
   Component C is the unique seclusion-maximal one satisfying the imposed conditions.
3. If there is a vertex set $U \subseteq V(G) \setminus (S \cup T)$ such that:
   • each connected component of $G[U]$ is adjacent to a vertex of S, and
   • the set U is a tight enrichment of S with respect to $\mathcal{F}$ (so $G[S \cup U]$ contains a new partial forbidden graph)
   then execute the following calls and stop:
   (a) For each $u \in U$ call $\mathsf{Enum}_{\mathcal{F}}(G - u, S, T, k - 1)$.
   (b) Call $\mathsf{Enum}_{\mathcal{F}}(G, S \cup U, T, k)$.
   A tight enrichment can have at most $||\mathcal{F}||$ vertices which bounds the branching factor in Step 3a. Note that these are exhaustive even though we do not consider adding U to T: since each component of $G[U]$ is adjacent to a vertex of S, if a relevant secluded subgraph does not contain all of U then it contains some vertex of U in its neighborhood and we find it in Step 3a. Below, we prove that some progress measure improves in all calls.
4. For the rest of the algorithm, let $\mathcal{U}$ denote the collection of all connected vertex sets $U \subseteq V(G) \setminus (S \cup T)$ which form tight enrichments of S with respect to $\mathcal{F}$. Let $V(\mathcal{U}) := \bigcup_{U \in \mathcal{U}} U$.
   (a) If $\lambda^{\mathsf{L}}_G(S, T) < \lambda^{\mathsf{L}}_G(S, T \cup V(\mathcal{U}))$: then (using Lemma 10) there exists $U \in \mathcal{U}$ such that $\lambda^{\mathsf{L}}_G(S, T) < \lambda^{\mathsf{L}}_G(S, T \cup U)$, execute the following calls and stop:
      i. For each $u \in U$ call $\mathsf{Enum}_{\mathcal{F}}(G - u, S, T, k - 1)$. (The value of k decreases.)
      ii. Call $\mathsf{Enum}_{\mathcal{F}}(G, S \cup U, T, k)$. (We absorb a new partial forbidden graph.)
      iii. Call $\mathsf{Enum}_{\mathcal{F}}(G, S, T \cup U, k)$. (The separator size increases.)
   (b) If $\lambda^{\mathsf{L}}_G(S, T) = \lambda^{\mathsf{L}}_G(S, T \cup V(\mathcal{U}))$, then let P be the farthest left-restricted minimum $(S, T \cup V(\mathcal{U}))$-separator in G; it exists since $\lambda^{\mathsf{L}}_G(S, T) = \lambda^{\mathsf{L}}_G(S, T \cup V(\mathcal{U})) \leq k$. Define $S' = R_G(S, P) \supseteq S$. Pick an arbitrary $p \in P$ (which may be contained in T but not in S).
      i. Call $\mathsf{Enum}_{\mathcal{F}}(G - p, S', T \setminus \{p\}, k - 1)$. (The value of k decreases.)
      ii. If $p \notin T$, then call $\mathsf{Enum}_{\mathcal{F}}(G, S' \cup \{p\}, T, k)$.
      (Either here or in the next recursive call we will be able to make progress.)
   It might happen that $\mathcal{U}$ is empty; in this case the algorithm will execute Step 4b. Also note that P is non-empty because the algorithm did not stop in Step 2; hence it is always possible to choose a vertex $p \in P$.

Before providing an in-depth analysis of the algorithm, we establish that it always terminates. For each recursive call, either a vertex outside S is deleted, or one of S or T grows in size while the two remain disjoint. Since S and T are vertex subsets of a finite graph, this process terminates. The key argument in the correctness of the algorithm is formalized in the following claim.

Claim 2. *If the algorithm reaches Step 4b, then every seclusion-maximal k-secluded subgraph satisfying the conditions of the theorem statement contains S'.*

Proof. We prove the claim by showing that for an arbitrary induced subgraph $G[C]$ that (a) is k-secluded, (b) is connected, (c) is $\mathcal{F}$-free, (d) contains all vertices of S, and (e) contains no vertex of T, the subgraph induced by $C \cup S'$ also satisfies (a)–(e) while $|N_G(C \cup S')| \leq |N_G(C)|$. Hence any seclusion-maximal subgraph satisfying the conditions contains S'.

Under the conditions of Step 4b, we have $\lambda^{\mathrm{L}}_G(S, T) = \lambda^{\mathrm{L}}_G(S, T \cup V(\mathcal{U}))$, implying that the set P is a left-restricted minimum (S, T)-separator. Next, we have $S' = R_G(S, P)$. By exploiting submodularity of the size of the open neighborhood, we proved in Lemma 11 that $|N_G(C \cup S')| \leq |N_G(C)|$. The key part of the argument is to prove that $C \cup S'$ induces an $\mathcal{F}$-free subgraph. Assume for a contradiction that $G[C \cup S']$ contains an induced subgraph isomorphic to $F \in \mathcal{F}$ and let $\phi \colon V(F) \to C \cup S'$ denote an induced subgraph isomorphism. Out of all ways to choose ϕ, fix a choice that minimizes the number of vertices $|\phi(F) \setminus S|$ the subgraph uses from outside S. We distinguish two cases.

Neighborhood of S intersects $\phi(F)$ If $\phi(F) \cap N_G(S) \neq \emptyset$, then we will use the assumption that Step 3 of the algorithm was not applicable to derive a contradiction. Let F' be the graph consisting of those connected components F_i of F for which $\phi(F_i) \cap N_G[S] \neq \emptyset$; let $U = \phi(F') \setminus S$. Observe that each connected component of $G[U]$ is adjacent to a vertex of S. By construction U is disjoint from S, and U is disjoint from T since $\phi(F) \subseteq C \cup S'$ while both these sets are disjoint from T. Hence U satisfies all but one of the conditions for applying Step 3. Since the algorithm reached Step 4b, it follows that U failed the last criterion which means that the partial forbidden graph F' also exists as an induced subgraph in $G[S]$. Let $\phi_{F'} \colon V(F') \to S$ be an induced subgraph isomorphism from F' to $G[S]$. Since all vertices $v \in V(F)$ for which $\phi(v) \in N_G[S]$ satisfy $v \in V(F')$, we can define a new subgraph isomorphism ϕ' of F in $G[C \cup S']$ as follows for each $v \in V(F)$:

$$\phi'(v) = \begin{cases} \phi_{F'}(v) & \text{if } v \in F' \\ \phi(v) & \text{otherwise.} \end{cases} \tag{1}$$

Observe that this is a valid induced subgraph isomorphism since F' consists of some connected components of F, and we effectively replace the image of F' by $\phi_{F'}$. Since the image of the remaining graph $\overline{F'} = F - F'$ does not use any vertex of $N_G[S]$ by definition of F', there are no edges between vertices of $\phi_{F'}(F')$ and vertices of $\phi(\overline{F'})$, which validates the induced subgraph isomorphism.

Since $\phi(F)$ contains at least one vertex from $N_G(S)$ while $\phi'(F)$ does not, and the only vertices of $\phi'(F) \setminus \phi(F)$ belong to S, we conclude that $\phi'(F)$ contains strictly fewer vertices outside S than $\phi(F)$; a contradiction to minimality of ϕ.

Neighborhood of S does not intersect $\phi(F)$ Now suppose that $\phi(F) \cap N_G(S) = \emptyset$. If $\phi(F) \subseteq C$, then $\phi(F)$ is an induced F-subgraph in $G[C]$, a contradiction to the assumption that C is $\mathcal{F}$-free. Hence $\phi(F)$ must contain a vertex $v \in S' \setminus C \subseteq S' \setminus S$. Since the neighborhood of S does not intersect $\phi(F)$, we have $v \notin N_G(S)$ and therefore $v \in S' \setminus N_G[S]$.

Fix an arbitrary connected component F_i of F for which $\phi(F_i)$ contains a vertex of $S' \setminus N_G[S]$. We derive several properties of $\phi(F_i)$.

(i) Since F_i is a connected component of F, the graph $G[\phi(F_i)]$ is connected.
(ii) We claim that $\phi(F_i) \cap S = \emptyset$. Note that a connected subgraph cannot both contain a vertex from S and a vertex outside $N_G[S]$ without intersecting $N_G(S)$. Since $\phi(F) \cap N_G(S) = \emptyset$ by the case distinction, the graph $G[\phi(F_i)]$ is connected since F_i is connected, and $\phi(F_i)$ contains a vertex of $S' \setminus N_G[S]$, we find $\phi(F_i) \cap S = \emptyset$.
(iii) We have $\phi(F_i) \cap T = \emptyset$, since $\phi(F) \subseteq C \cup S'$ while both C and S' are disjoint from T.
(iv) We claim that $\phi(F_i) \notin \mathcal{U}$. To see that, recall that $S' = R_G(S, P)$ is the set of vertices reachable from S when removing the $(S, T \cup V(\mathcal{U}))$-separator P. The definition of separator P therefore ensures that no vertex of S' belongs to $V(\mathcal{U})$. Since $\phi(F_i)$ contains a vertex of $S' \setminus N_G[S]$ by construction, some vertex of $\phi(F_i)$ does not belong to $V(\mathcal{U})$ and therefore $\phi(F_i) \notin \mathcal{U}$.

Now note that $\phi(F_i)$ satisfies almost all requirements for being contained in the set $\mathcal{U}$ defined in Step 4: it induces a connected subgraph by Property (i) and it is disjoint from $S \cup T$ by Properties (ii)–(iii). From Property (iv), stating that $\phi(F_i) \notin \mathcal{U}$, we therefore conclude that it fails the last criterion: the set $\phi(F_i)$ is not a tight enrichment of S.

Let F' be the graph formed by F_i together with all components F_j of F for which $\phi(F_j) \subseteq S$; then $\phi(F_i) = \phi(F') \setminus S$. Note that F' is a partial forbidden graph. Let us consider why $\phi(F_i)$ fails to be a tight enrichment of S. Note that the restriction $\phi_{|V(F')}$ of ϕ to the domain $V(F')$ is an induced subgraph isomorphism from the partial forbidden graph F' to G and that $\phi_{|V(F')}(F_i) = \phi(F_i)$. Since $\phi_{|V(F')}(F_i) = \phi_{|V(F')}(F') \setminus S$, the set $\phi(F_i)$ satisfies the conditions of Definition 13 that an enrichment should satisfy in order to be tight. Since $\phi(F_i)$ is not a tight enrichment, it follows that $\phi(F_i)$ is not an enrichment at all. This means that all partial forbidden graphs that occur as induced subgraph in $G[S \cup \phi(F_i)]$, also occur in $G[S]$. Since $\phi(F_i) \cup S$ contains $\phi(F')$, we know F' is isomorphic to an induced subgraph of $G[S \cup \phi(F_i)]$. By the previous argument, the graph F' therefore also occurs as induced subgraph of $G[S]$. Let $\phi_{F'} \colon F' \to S$ denote an induced subgraph isomorphism of F' to $G[S]$. Since $\phi(F)$ contains no vertex of $N_G(S)$, we can define a new subgraph isomorphism ϕ' of F in $G[C \cup S']$ exactly as in Equation (1).

As above, let $\overline{F'} = F - F'$. Since the graph F' consists of some connected components of F, while $\phi_{F'}(F') \subseteq S$ and $\phi(\overline{F'}) \cap N_G[S] = \emptyset$, it follows that ϕ' is an induced subgraph isomorphism of F in $G[C \cup S']$. But $|\phi'(F) \setminus S|$ is strictly smaller than $|\phi(F) \setminus S|$ since $\phi(F_i)$ intersects $S' \setminus N_G[S]$ while $\phi'(F_i) \subseteq \phi'(F') \subseteq S$ and ϕ and ϕ' coincide on $\overline{F'}$. This contradicts the minimality of the choice of ϕ.

Since the case distinction is exhaustive, this proves the claim. ⌟

Using the previous claim, we can establish the correctness of the algorithm.

Claim 3. *If $G[C]$ is an induced subgraph of G that is seclusion-maximal with respect to being connected, $\mathcal{F}$-free, k-secluded and satisfying $S \subseteq C \subseteq V(G) \setminus T$, then C occurs in the output of* $\mathsf{Enum}_{\mathcal{F}}(G, S, T, k)$.

Proof. We prove this claim by induction on the recursion depth of the $\mathsf{Enum}_{\mathcal{F}}$ algorithm, which is valid as we argued above it is finite. In the base case, the algorithm does not recurse. In other words, the algorithm either stopped in Step 1 or 2. If the algorithm stops in Step 1, then there can be no induced subgraph satisfying the conditions and so there is nothing to show. If the algorithm stops in Step 2, then the only seclusion-maximal induced subgraph is the $\mathcal{F}$-free connected component containing S. Note that this component is k-secluded since $k \geq 0$ as $\lambda^{\mathsf{L}}_G(S,T) \geq 0$ and the algorithm did not stop in Step 1a.

For the induction step, we may assume that each recursive call made by the algorithm correctly enumerates a superset of the seclusion-maximal subgraphs satisfying the conditions imposed by the parameters of the recursive call, as the recursion depth of the execution of those calls is strictly smaller than the recursion depth for the current arguments (G, S, T, k). Consider a connected $\mathcal{F}$-free k-secluded induced subgraph $G[C]$ of G with $S \subseteq C \subseteq V(G) \setminus T$ that is seclusion-maximal with respect to satisfying all these conditions. Suppose there is a vertex set $U \subseteq V(G) \setminus (S \cup T)$ that satisfies the conditions of Step 3. If $U \subseteq C$, then by induction C is part of the enumerated output of Step 3b. Otherwise, since each connected component of $G[U]$ is adjacent to a vertex in S, there is at least one vertex $u \in U$ such that $u \in N_G(C)$. By Observation 14, the output of the corresponding call in Step 3a contains C. Note that since $U \cap T = \emptyset$, we have $T \subseteq V(G) \setminus (S \cup U)$ and therefore the recursive calls satisfy the input requirements.

Next we consider the correctness in case there is no set U that satisfies the conditions of Step 3, so that the algorithm reaches Step 4. Let $\mathcal{U}$ be the set of tight enrichments as defined in Step 4. First suppose that $\lambda^{\mathsf{L}}_G(S,T) < \lambda^{\mathsf{L}}_G(S, T \cup V(\mathcal{U}))$. Then by the contrapositive of the first part of Lemma 10 with $Z = V(\mathcal{U})$, there is a vertex $v \in V(\mathcal{U}) \setminus T$ such that $\lambda^{\mathsf{L}}_G(S, T \cup \{v\}) > \lambda^{\mathsf{L}}_G(S,T)$. By picking an enrichment $U \in \mathcal{U}$ such that $v \in U$, this implies $\lambda^{\mathsf{L}}_G(S, T \cup U) > \lambda^{\mathsf{L}}_G(S,T)$. Now if there is a vertex $u \in U$ such that $u \in N_G(C)$, then by induction and Observation 14 we get that C is output by the corresponding call in Step 4(a)i. Otherwise, either $U \subseteq C$ or $U \cap C = \emptyset$ (since U is connected) and C is found in Step 4(a)ii or Step 4(a)iii respectively. Again observe that these recursive calls satisfy the input requirements as $U \cap (S \cup T) = \emptyset$.

Finally suppose that $\lambda^{\mathsf{L}}_G(S,T) = \lambda^{\mathsf{L}}_G(S, T \cup V(\mathcal{U}))$. By Claim 2 we get that $S' \subseteq C$. We first argue that $P = N_G(S')$ is non-empty. Note that since the algorithm did not stop in Step 1, the graph $G[S]$ is $\mathcal{F}$-free and S is contained in a single connected component of G. Furthermore since it did not stop in Step 2, the connected component containing S either has a vertex of T or is not $\mathcal{F}$-free. Note that the former case already implies $\lambda^{\mathsf{L}}_G(S,T) > 0$, so that P is non-empty. If the component of G that contains S has no vertex of T and is not $\mathcal{F}$-free, then it contains a vertex set J for which $G[J]$ is isomorphic to some $F \in \mathcal{F}$. Observe that $J \setminus (S \cup T) = J \setminus S$ is a tight enrichment of S. We have established that it is possible to enrich S, but to relate to the condition of Step 4 we need an enrichment that is connected. Let $U \subseteq J$ be a minimum-size tight enrichment contained in J; possibly $U = J$. Let $\phi \colon V(F') \to V(G)$ be the corresponding subgraph isomorphism from some partial forbidden graph F'; we have $U = \phi(F') \setminus S$. Note that all vertices of $U \subseteq J$ belong to the same connected component of G as all vertices of S. We argue that $G[U]$ is connected. If each connected component of $G[U]$ is adjacent to a vertex of S, then Step 3 would have applied, contradicting the fact that the algorithm reaches Step 4. Hence, there exists a connected component of $G[U]$ that is non-adjacent to S; let U' be the vertex set of such a component. Since U is chosen to be minimum, we get that $U \setminus U'$ is not a tight enrichment, and so there is an induced subgraph of $G[S]$ isomorphic to the partial forbidden graph $F'' = G[\phi(F') \setminus U']$. This subgraph of $G[S]$ combines with the graph $G[U']$ to form an induced subgraph isomorphic to F' (we exploit that U' is not adjacent to S), which shows that U' is a tight enrichment. By minimality of U we obtain $U = U'$. Hence U is not adjacent to S and the graph $G[U]$ is connected so $U \in \mathcal{U}$. Since U and S are contained in the same connected component of G we get that $\lambda^{\mathsf{L}}_G(S, T \cup V(\mathcal{U})) > 0$. This implies there exists some vertex $p \in P = N_G(S')$. Since $S' \subseteq C$, we either get $p \in N_G(C)$, or (if $p \notin T$) $p \in C$. By induction (and Observation 14) we conclude that C is part of the output of Step 4(b)i or Step 4(b)ii. The condition $p \notin T$ ensures that the input requirements of the latter recursive call are satisfied. ⌟

As the previous claim shows that the algorithm enumerates a superset of the relevant seclusion-maximal induced subgraphs, to prove Theorem 2 it suffices to bound the size of the search tree generated by the algorithm, and thereby the running time and total number of induced subgraphs which are given as output. To that end, we argue that for any two successive recursive calls in the recursion tree, at least one of them makes strict progress on a relevant measure. Since no call can increase the measure, this will imply a bound on the depth of the recursion tree. Since it is easy to see that the branching factor is a constant depending on $||\mathcal{F}||$, this will lead to the desired bound.

Claim 4. *The search tree generated by the call* $\mathsf{Enum}_{\mathcal{F}}(G, S, T, k)$ *has depth* $\mathcal{O}_{\mathcal{F}}(k)$ *and* $2^{\mathcal{O}_{\mathcal{F}}(k)}$ *leaves.*

Proof. Let $g(X)$ denote the number of partial forbidden graphs of $\mathcal{F}$ that appear as an induced subgraph in $G[X]$; note that $g(X) \leq \sum_{F \in \mathcal{F}} 2^{|V(F)|}$. For the running time analysis, we consider the progress measure $k + (k - \lambda^{\mathrm{L}}_G(S, T)) + (g(V(G)) - g(S))$. We argue that the measure drops by at least one after two consecutive recursive calls to the algorithm. For most cases, the measure already drops in the first recursive call. First suppose that a recursive call is made in Step 3a, then the third summand does not increase: S does not change while $g(V(G) \setminus \{u\}) \leq g(V(G))$. We have $\lambda^{\mathrm{L}}_{G-u}(S, T) \geq \lambda^{\mathrm{L}}_G(S, T) - 1$. Since k is decreased by one, the measure strictly goes down. Next suppose a recursive call is made in Step 3b. Since $g(S \cup U) > g(S)$ by construction, and $\lambda^{\mathrm{L}}_G(S \cup U, T) \geq \lambda^{\mathrm{L}}_G(S, T)$, again the measure strictly goes down. The fact that the measure drops for a recursive call in Step 4(a)i follows akin to the arguments for Step 3a. The same holds for Step 4(a)ii akin to Step 3b. For a recursive call made in Step 4(a)iii, we know by assumption that $\lambda^{\mathrm{L}}_G(S, T \cup U) > \lambda^{\mathrm{L}}_G(S, T)$. Since k and S remain the same, the measure strictly decreases.

The reasoning becomes more involved for a recursive call in Step 4b. For a recursive call in Step 4(b)i, we have $g(S') \geq g(S)$ as $S \subseteq S'$, while $\lambda^{\mathrm{L}}_{G-p}(S', T \setminus \{p\}) = \lambda^{\mathrm{L}}_G(S, T) - 1$ since p belongs to a minimum left-restricted (S, T)-separator in G, which is also a left-restricted minimum (S', T)-separator. Since k goes down by one, the measure strictly decreases.

Finally, consider a recursive call made in Step 4(b)ii (so $p \notin T$). Note that $g(S' \cup \{p\}) \geq g(S)$ as $S \subseteq S'$, k remains the same, and $\lambda^{\mathrm{L}}_G(S' \cup \{p\}, T) \geq \lambda^{\mathrm{L}}_G(S, T)$. We distinguish three cases, depending on whether p is in some enrichment, and show that in each case we make progress in the recursive call because the measure $k + (k - \lambda^{\mathrm{L}}_G(S, T)) + (g(V(G)) - g(S))$ decreases.

- If $\{p\} \in \mathcal{U}$, then actually $g(S' \cup \{p\}) > g(S)$ and the measure strictly drops.
- If $\{p\}$ is not a tight enrichment of S, but $p \in U$ for some $U \in \mathcal{U}$, observe that $U \setminus \{p\}$ is disjoint from $S' \cup \{p\} \cup T$, forms a tight enrichment of $S' \cup \{p\}$, and each connected component of $G[U \setminus \{p\}]$ is adjacent to $p \in S' \cup \{p\}$ as $G[U]$ is connected. It follows that in the next call Step 3 applies (which it reaches as we assumed the algorithm recurses twice) and again we make progress.
- In the remainder we have $p \notin V(\mathcal{U})$. First consider the case that $\lambda^{\mathrm{L}}_G(S' \cup \{p\}, T) = \lambda^{\mathrm{L}}_G(S' \cup \{p\}, T \cup V(\mathcal{U}))$. Then since $P = N_G(S')$ was a farthest $(S, T \cup V(\mathcal{U}))$-separator, by Lemma 9(3) we get that $\lambda^{\mathrm{L}}_G(S' \cup \{p\}, T) = \lambda^{\mathrm{L}}_G(S' \cup \{p\}, T \cup V(\mathcal{U})) > \lambda^{\mathrm{L}}_G(S, T \cup V(\mathcal{U})) = \lambda^{\mathrm{L}}_G(S, T)$, and therefore the progress measure strictly drops.
  In the remaining case we have $\lambda^{\mathrm{L}}_G(S' \cup \{p\}, T) < \lambda^{\mathrm{L}}_G(S' \cup \{p\}, T \cup V(\mathcal{U}))$. Since $p \notin V(\mathcal{U})$, if the algorithm reaches Step 4 in the next recursive call, the set of enrichments $\mathcal{U}$ remains the same. But then Step 4a applies, which makes progress in the measure as argued above.

We have shown that the measure decreases by at least one after two consecutive recursive calls. The algorithm cannot proceed once the measure becomes negative because $g(S)$ cannot grow beyond $g(V(G))$ and whenever $k < 0$ or $\lambda^{\mathrm{L}}_G(S, T) > k$ the algorithm immediately stops. Since $g(V(G))$ is upper-bounded by a constant depending on $||\mathcal{F}||$ and $|\mathcal{F}|$, we infer that the search tree has depth $\mathcal{O}_{\mathcal{F}}(k)$. Any tight enrichment detected in Step 3 or Step 4 can have at most $||\mathcal{F}||$ vertices, so the branching factor is bounded by $||\mathcal{F}||$. Hence, the search tree has $2^{\mathcal{O}_{\mathcal{F}}(k)}$ leaves as required. ⌟

The previous claim implies that the number of seclusion-maximal connected $\mathcal{F}$-free k-secluded induced subgraphs containing all of S and none of T is $2^{\mathcal{O}_{\mathcal{F}}(k)}$, since the algorithm outputs at most one subgraph per call and only does so in leaf nodes of the recursion tree. As Claim 4 bounds the size of the search tree generated by the algorithm, the desired bound on the total running time follows from the claim below.

Claim 5. *A single iteration of* $\mathsf{Enum}_{\mathcal{F}}(G, S, T, k)$ *can be implemented to run in time* $|\mathcal{F}| \cdot 2^{||\mathcal{F}||} \cdot n^{||\mathcal{F}|| + \mathcal{O}(1)}$ *and polynomial space.*

Proof. Within this proof, for a graph F we abbreviate $|V(F)|$ to $|F|$. Deciding whether $\lambda^{\mathrm{L}}_G(S, T) > k$, as required in Step 1, can be done in $\mathcal{O}(k(n+m))$ time by Theorem 6 and Observation 7. Finding the connected components of G, and deciding if S is contained in only one can be done in $\mathcal{O}(n+m)$ time. Deciding if $G[S]$ contains an induced subgraph isomorphic to some $F \in \mathcal{F}$ can be done in $|\mathcal{F}| \cdot n^{||\mathcal{F}|| + \mathcal{O}(1)}$ time. In the same running time we can decide if the connected component containing S is $\mathcal{F}$-free and contains nothing of T as needed for Step 2.

For Step 3, we proceed as follows. For each $F \in \mathcal{F}$, for each partial forbidden graph F' of F (which consists of some subset of the connected components of F), verify whether there is an induced subgraph of $G[S]$ isomorphic to F' by checking all of the at most $n^{|F'|}$ ways in which it could appear and verifying in $\mathcal{O}(n^2)$ time if the right adjacencies are there. Keep track of which partial forbidden graphs are not present in $G[S]$. Next, for each partial forbidden graph F' not appearing in

$G[S]$, for each of the at most $n^{|F'|}$ induced subgraph isomorphisms $\phi: V(F') \to V(G) \setminus T$ we verify whether each connected component of $U = \phi(F') \setminus S$ is adjacent to a vertex of S. This brings the total time for Step 3 to $|\mathcal{F}| \cdot 2^{||\mathcal{F}||} \cdot n^{||\mathcal{F}||+\mathcal{O}(1)}$.

In the same time we can compute $\mathcal{U}$ for Step 4 (this time, $G[U]$ should be connected rather than each component being adjacent to S). Then, deciding if $\lambda^{\mathrm{L}}_G(S,T) < \lambda^{\mathrm{L}}_G(S, T \cup V(\mathcal{U}))$ for Step 4a can be done in $\mathcal{O}(k(n+m))$ time by Theorem 6 and Observation 7 since $\lambda^{\mathrm{L}}_G(S,T) \leq k$. Finding $U \in \mathcal{U}$ such that $\lambda^{\mathrm{L}}_G(S, T \cup U) > \lambda^{\mathrm{L}}_G(S,T)$ can be done in $\mathcal{O}(n^{||\mathcal{F}||} \cdot k(n+m))$ time. If Step 4a does not apply, then automatically Step 4b does and so we get $\lambda^{\mathrm{L}}_G(S,T) = \lambda^{\mathrm{L}}_G(S, T \cup V(\mathcal{U}))$. Finally, computing the farthest left-restricted minimum $(S, T \cup V(\mathcal{U}))$-separator can be done in $\mathcal{O}(k(n+m))$ time by Lemma 9. It is easy to see that the steps above can be carried out using polynomial space. ⌟

With the claims above, we can wrap up the proof of Theorem 2. Claim 3 shows (aided by Claim 2) that the output of the enumeration algorithm contains all seclusion-maximal $\mathcal{F}$-free subgraphs C that satisfy the conditions from the theorem statement. Claim 4 establishes that the search tree generated by a call $\mathsf{Enum}_{\mathcal{F}}(G,S,T,k)$ of the recursive algorithm has depth $\mathcal{O}_{\mathcal{F}}(k)$ and $2^{\mathcal{O}_{\mathcal{F}}(k)}$ leaves. It therefore has $2^{\mathcal{O}_{\mathcal{F}}(k)} \cdot k \in 2^{\mathcal{O}_{\mathcal{F}}(k)}$ nodes in total. By Claim 5, a single iteration can be implemented to run in time $|\mathcal{F}| \cdot 2^{||\mathcal{F}||} \cdot n^{||\mathcal{F}||+\mathcal{O}(1)}$ (not counting the time spent in recursive calls), using polynomial space. The total running time is bounded by the product of number of nodes in the recursion tree and the time per node, hence by $2^{\mathcal{O}_{\mathcal{F}}(k)} \cdot |\mathcal{F}| \cdot 2^{||\mathcal{F}||} \cdot n^{||\mathcal{F}||+\mathcal{O}(1)} \in 2^{\mathcal{O}_{\mathcal{F}}(k)} \cdot n^{||\mathcal{F}||+\mathcal{O}(1)}$. This concludes the proof of Theorem 2. □

We conclude the section by a remark on the running time of the algorithm in Theorem 2. One might be interested to know a bound on the running time without $\mathcal{O}_{\mathcal{F}}$ notation, instead using only the standard $\mathcal{O}$-notation that suppresses universal constants but does not suppress factors depending on $\mathcal{F}$. The analysis of Claim 4 shows that the recursion tree generated by the algorithm has a branching factor of $\mathcal{O}(||\mathcal{F}||)$ and a depth of $\mathcal{O}(k + \sum_{F \in \mathcal{F}} 2^{|V(F)|})$. The number of nodes in the recursion tree can therefore be bounded by $\mathcal{O}(||\mathcal{F}||)^{\mathcal{O}(k)} \cdot \mathcal{O}(||\mathcal{F}||)^{\mathcal{O}(\sum_{F \in \mathcal{F}} 2^{|V(F)|})}$. Multiplying this expression with the bound from Claim 5 on the time per iteration gives an alternative upper-bound on the running time, which can be rewritten as $|\mathcal{F}| \cdot \mathcal{O}(||\mathcal{F}||)^{\mathcal{O}(\sum_{F \in \mathcal{F}} 2^{|V(F)|})} \cdot 2^{\mathcal{O}(k \log ||\mathcal{F}||)} \cdot n^{||\mathcal{F}||+\mathcal{O}(1)}$.

4. Applications

As applications of Theorem 2, we derive faster algorithms for two problems studied in the literature. The first problem is formally defined as follows [32] for any finite set $\mathcal{F}$ of undirected graphs.

Connected k-Secluded $\mathcal{F}$-Free Subgraph	**Parameter:** k
Input: Graph G, integer k, weight function $w\colon V(G) \to \mathbb{Z}_{>0}$.	
Task: Find a connected k-secluded set $C \subseteq V(G)$ for which $G[C]$ is $\mathcal{F}$-free which maximizes $\sum_{v \in C} w(v)$.	

A single-exponential algorithm for this problem follows easily from Corollary 3.

Corollary 4. *For each fixed finite family $\mathcal{F}$, Connected k-Secluded $\mathcal{F}$-Free Subgraph can be solved in time $2^{\mathcal{O}_{\mathcal{F}}(k)} \cdot n^{||\mathcal{F}||+\mathcal{O}(1)}$ and polynomial space.*

Proof. Since the weights are positive, any maximum-weight solution to the problem is seclusion-maximal with respect to being k-secluded, connected, and $\mathcal{F}$-free. We can therefore solve an instance (G,k,w) as follows. Invoke Corollary 3 to enumerate a superset of the all seclusion-maximal connected $\mathcal{F}$-free k-secluded induced subgraphs containing $S := \{v\}$. For each enumerated set C, check whether it is indeed $\mathcal{F}$-free in time $n^{||\mathcal{F}||+\mathcal{O}(1)}$. The heaviest, taken over all choices of v and C, is given as the output. □

Our second application concerns deletion problems to scattered graph classes, which are defined for finite sequences $(\Pi_1, \ldots, \Pi_d)$ of graph classes.

$(\Pi_1, \ldots, \Pi_d)$-Deletion	**Parameter:** k
Input: Graph G and integer k.	
Question: Is there a vertex set $X \subseteq V(G)$ of size at most k, such that for each connected component C of $G - X$ there exists $i \in [d]$ such that $C \in \Pi_i$?	

By exploiting the fact that each connected component of $G - X$ is k-secluded, we can obtain single-exponential FPT algorithms for this problem when each graph class Π is characterized by a finite number of forbidden induced subgraphs. In the following statement, both $\mathcal{O}_\Pi$'s hide factors depending on the choice of $(\Pi_1, \ldots, \Pi_d)$.

Theorem 5. *$(\Pi_1, \ldots, \Pi_d)$-Deletion can be solved in time $2^{\mathcal{O}_\Pi(k)} \cdot n^{\mathcal{O}_\Pi(1)}$ and polynomial space when each graph class Π_i is characterized by a finite set $\mathcal{F}_i$ of (not necessarily connected) forbidden induced subgraphs.*

Proof. We describe an algorithm for the problem. If $k < 0$, report that it is a no-instance. If there is a connected component that belongs to Π_i for some $i \in [d]$, then delete the component and continue ([34, Reduction Rule 1]). If the graph becomes empty, return that it is a yes-instance. Otherwise, if $k = 0$, report that it is a no-instance.

In the remainder we have $k > 0$ and G non-empty. Pick a vertex $v \in V(G)$. There are two cases; v either belongs to the solution set X, or belongs to a component of $G - X$ that is contained in some graph class Π_i. We perform branching to cover both options. For the first option, recursively call the algorithm on $G - v$ searching for a solution of size $k - 1$. For the second option, for each $i \in [d]$ and $s \in [k]$, invoke Theorem 2 to enumerate (a superset of) the seclusion-maximal connected $\mathcal{F}_i$-free s-secluded subgraphs containing v. Note that the theorem implies this output has at most c^s elements for some constant c. For each of the enumerated subgraphs C such that $G[C] \in \Pi_i$ and $|N_G(C)| = s$, recursively call the algorithm on $G - N_G[C]$ searching for a solution of size $k - |N_G(C)|$. Output yes if and only if one of the recursive calls results in a yes-instance.

For correctness of the algorithm, we argue that the enumeration of seclusion-maximal secluded subgraphs suffices. Suppose there is a solution X not containing v such that the component C containing v in $G - X$ belongs to Π_i. If C was among the output of the enumeration algorithm, it is easy to see that the algorithm is correct. Suppose that C was not enumerated because it is not seclusion-maximal. For this choice of i and $s = |N_G(C)|$, the enumeration included some connected $\mathcal{F}_i$-free s-secluded subgraph C' with $C \subseteq C'$ and $|N_G(C')| \leq |N_G(C)|$. Since the target graph classes are hereditary and graph $G - N_G[C]$ admits solution $X \setminus N_G(C)$ of size at most $k - |N_G(C)|$, then its induced subgraph $G - N_G[C']$ admits a solution X' of size at most $k - |N_G(C')|$. Hence, $X' \cup N_G(C')$ is also a valid solution for G of size at most k. We conclude that the branching algorithm always finds a solution if there is one.

We turn to the running time. Let $T(k)$ denote the number of leaves in the recursion tree for a call with parameter k, where $T(0) = 1$. By grouping the secluded subgraphs by their neighborhood size, observe that T satisfies $T(k) = T(k - 1) + d \cdot \sum_{i=1}^{k} c^i \cdot T(k-i) \leq (d+1) \cdot \sum_{i=1}^{k} c^i \cdot T(k-i)$ (the inequality clearly holds if $c \geq 1$). By induction we argue that $T(k) \leq ((d+1)2c)^k$, which trivially holds if $k = 0$. Suppose that it holds for all values below k; then we derive:

$$
\begin{aligned}
T(k) &\leq (d+1) \cdot \sum_{i=1}^{k} c^i \cdot T(k-i) && \text{By grouping on neighborhood size.}\\
&\leq (d+1) \cdot \sum_{i=1}^{k} c^i \cdot ((d+1)2c)^{k-i} && \text{By induction.}\\
&\leq ((d+1)c)^k \cdot \sum_{i=1}^{k} 2^{k-i} && \text{Using } (d+1)^{k-i} \leq (d+1)^{k-1}.\\
&\leq ((d+1)2c)^k. && \text{Since } \sum_{i=0}^{k-1} 2^i < 2^k.
\end{aligned}
$$

Since the depth of the recursion tree is at most k, the recursion tree has at most $k \cdot ((d+1)2c)^k$ nodes. Finally we consider the running time per node of the recursion tree. Finding the connected components can be done in $\mathcal{O}(n+m)$ time. Checking if one of them belongs to Π_i for some $i \in [d]$ can be done in $n^{\mathcal{O}_\Pi(1)}$ time. The time needed for the $d \cdot k$ calls to Theorem 2 is $dk \cdot 2^{\mathcal{O}_\Pi(k)} \cdot n^{\mathcal{O}_\Pi(1)}$. Since d and c are constants, we get the claimed running time.

Note that since Theorem 2 uses polynomial space, and we process its output one at a time without storing it, we conclude that the described algorithm uses polynomial space. □

5. Conclusion

We have introduced a new algorithmic primitive based on secluded connected subgraphs which generalizes important separators. The high-level idea behind the algorithm is *enumeration via separation*: by introducing an artificial set T and considering the more general problem of enumerating secluded subgraphs containing S but disjoint from T, we can analyze the progress of the recursion in terms of the size of a minimum (left-restricted) (S, T)-separator. We expect this idea to be useful in scenarios beyond the one studied here.

We presented a single-exponential, polynomial-space FPT algorithm to enumerate the family of seclusion-maximal connected $\mathcal{F}$-free subgraphs for finite $\mathcal{F}$, making it potentially viable for practical use [39]. The combination of single-exponential running time and polynomial space usage sets our approach apart from others such as recursive understanding [16–18] and treewidth reduction [4]. Algorithms exploiting half-integrality of the linear-programming relaxation or other discrete relaxations also have these desirable properties, though [11–14,40]. Using this approach, Iwata, Yamaguchi, and Yoshida [14] even obtained a *linear-time* algorithm in terms of the number of vertices n, solving (vertex) Multiway Cut in time $2^k \cdot k \cdot (n+m)$. At a high level, there is some resemblance between their approach and ours. They work on a discrete relaxation of deletion problems in graphs which are not standard LP-relaxations, but are based on relaxations of a *rooted* problem in which only constraints involving a prescribed set S are active. This is reminiscent of the fact that we enumerate secluded subgraphs containing a prescribed set S. Their branching algorithms are based on the notion of an extremal optimal solution to the LP relaxation, which resembles our use of the farthest minimum left-restricted (S,T)-separator. However, the two approaches diverge there. To handle problems via their approach, they should be expressible as a 0/1/ALL CSP. Problems for which the validity of a solution can be verified by unit propagation (such as Node Unique Label Cover, Node Multiway Cut, Subset and Group Feedback Vertex Set) belong to this category, but it seems impossible to express the property of being $\mathcal{F}$-free for arbitrary finite sets $\mathcal{F}$ in this framework.

The branching steps underlying our algorithm were informed by the structure of the subgraphs induced by certain vertex sets. In the considered setting, where certain possibly disconnected structures are not allowed to appear inside C, it is necessary to characterize the forbidden sets in terms of the graph structure they induce. But when the forbidden sets are connected, we believe our proof technique can be used in a more general setting to establish the following. For any n-vertex graph G, non-empty vertex set $S \subseteq V(G)$, potentially empty $T \subseteq V(G) \setminus S$, integer k, and collection $F_1, \dots, F_m \subseteq V(G)$ of vertex sets of size at most ℓ which are connected in G, the number of k-secluded induced subgraphs $G[C]$ which are seclusion-maximal with respect to being connected, not containing any set F_i, and satisfying $S \subseteq C \subseteq V(G) \setminus T$, is bounded by $(2+\ell)^{\mathcal{O}(k)}$, and a superset of them can be enumerated in time $(2+\ell)^{\mathcal{O}(k)} \cdot m \cdot n^{\mathcal{O}(1)}$ and polynomial space. The reason why dealing with general connected obstacles is feasible is that whenever $F_i \cap C \neq \emptyset$ then also $F_i \cap N(C) \neq \emptyset$; this allows us to always make progress using the simpler branching strategy without keeping track of partial forbidden graphs. The corresponding generalization for *disconnected* vertex sets F_i is false, even for $|F_i| = 2$. To see this, consider a graph consisting of a cycle on $2m+1$ vertices consecutively labeled $s, a_1, \dots, a_m, b_1, \dots, b_m$ with $F_i = \{a_i, b_i\}$ for each $i \in [m]$, in which the number of relevant seclusion-maximal 2-secluded sets containing s is $\Omega(m)$.

We leave it to future work to consider generalizations of our ideas to *directed graphs*. Since important separators also apply in that setting, we expect the branching step in terms of left-restricted minimum separators to be applicable in directed graphs as well. However, there are multiple ways to generalize the notion of a connected secluded induced subgraph to the directed setting: one can consider weak connectivity, strong connectivity, or a rooted variant where we consider all vertices reachable from a source vertex x. Similarly, one can define seclusion in terms of the number of in-neighbors, out-neighbors, or both.

CRediT authorship contribution statement

Bart M.P. Jansen: Writing – review & editing, Writing – original draft, Formal analysis, Conceptualization. **Jari J.H. de Kroon:** Writing – review & editing, Formal analysis. **Michał Włodarczyk:** Writing – review & editing, Writing – original draft, Formal analysis, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 803421, ReduceSearch).

erc
European Research Council
Established by the European Commission

Data availability

No data was used for the research described in the article.

References

[1] L.R. Ford, D.R. Fulkerson, Maximal flow through a network, Can. J. Math. 8 (1956) 399–404, https://doi.org/10.4153/CJM-1956-045-5.
[2] D. Marx, Parameterized graph separation problems, Theor. Comput. Sci. 351 (3) (2006) 394–406, https://doi.org/10.1016/j.tcs.2005.10.007.
[3] N. Bousquet, J. Daligault, S. Thomassé, Multicut is FPT, SIAM J. Comput. 47 (1) (2018) 166–207, https://doi.org/10.1137/140961808.
[4] D. Marx, I. Razgon, Fixed-parameter tractability of multicut parameterized by the size of the cutset, SIAM J. Comput. 43 (2) (2014) 355–388, https://doi.org/10.1137/110855247.
[5] K. Kawarabayashi, M. Thorup, The minimum k-way cut of bounded size is fixed-parameter tractable, in: R. Ostrovsky (Ed.), IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS 2011, Palm Springs, CA, USA, October 22-25, 2011, IEEE Computer Society, 2011, pp. 160–169.

[6] M. Cygan, D. Lokshtanov, M. Pilipczuk, M. Pilipczuk, S. Saurabh, Minimum bisection is fixed-parameter tractable, SIAM J. Comput. 48 (2) (2019) 417–450, https://doi.org/10.1137/140988553.
[7] J. Chen, Y. Liu, S. Lu, B. O'Sullivan, I. Razgon, A fixed-parameter algorithm for the directed feedback vertex set problem, J. ACM 55 (5) (2008) 21:1–21:19, https://doi.org/10.1145/1411509.1411511.
[8] I. Razgon, B. O'Sullivan, Almost 2-SAT is fixed-parameter tractable, J. Comput. Syst. Sci. 75 (8) (2009) 435–450, https://doi.org/10.1016/j.jcss.2009.04.002.
[9] D. Marx, B. O'Sullivan, I. Razgon, Finding small separators in linear time via treewidth reduction, ACM Trans. Algorithms 9 (4) (2013) 30:1–30:35, https://doi.org/10.1145/2500119.
[10] D. Marx, Important separators and parameterized algorithms, in: P. Kolman, J. Kratochvíl (Eds.), Graph-Theoretic Concepts in Computer Science - 37th International Workshop, Revised Papers, vol. 6986, WG 2011, Teplá Monastery, Czech Republic, June 21-24, 2011, Lecture Notes in Computer Science., Springer, 2011, pp. 5–10.
[11] M. Cygan, M. Pilipczuk, M. Pilipczuk, J.O. Wojtaszczyk, On multiway cut parameterized above lower bounds, ACM Trans. Comput. Theory 5 (1) (2013) 3:1–3:11, https://doi.org/10.1145/2462896.2462899.
[12] S. Guillemot, FPT algorithms for path-transversal and cycle-transversal problems, Discrete Optim. 8 (1) (2011) 61–71, https://doi.org/10.1016/j.disopt.2010.05.003.
[13] Y. Iwata, M. Wahlström, Y. Yoshida, Half-integrality, LP-branching, and FPT algorithms, SIAM J. Comput. 45 (4) (2016) 1377–1411, https://doi.org/10.1137/140962838.
[14] Y. Iwata, Y. Yamaguchi, Y. Yoshida, 0/1/all CSPs, half-integral A-path packing, and linear-time FPT algorithms, in: M. Thorup (Ed.), 59th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2018, Paris, France, October 7-9, 2018, IEEE Computer Society, 2018, pp. 462–473.
[15] N. Misra, Kernelization, Planar F-deletion, in: Encyclopedia of Algorithms, Springer, 2016, pp. 1033–1036.
[16] R. Chitnis, M. Cygan, M. Hajiaghayi, M. Pilipczuk, M. Pilipczuk, Designing FPT algorithms for cut problems using randomized contractions, SIAM J. Comput. 45 (4) (2016) 1171–1229, https://doi.org/10.1137/15M1032077.
[17] M. Cygan, P. Komosa, D. Lokshtanov, M. Pilipczuk, M. Pilipczuk, S. Saurabh, M. Wahlström, Randomized contractions meet lean decompositions, ACM Trans. Algorithms 17 (1) (2021) 6:1–6:30, https://doi.org/10.1145/3426738.
[18] D. Lokshtanov, M.S. Ramanujan, S. Saurabh, M. Zehavi, Reducing CMSO model checking to highly connected graphs, in: I. Chatzigiannakis, C. Kaklamanis, D. Marx, D. Sannella (Eds.), 45th International Colloquium on Automata, Languages, and Programming, ICALP, 2018, July 9-13, 2018, Prague, Czech Republic, in: LIPIcs, vol. 107, Schloss Dagstuhl - Leibniz-Zentrum Für Informatik, 2018, pp. 135:1–135:14.
[19] E.J. Kim, S. Kratsch, M. Pilipczuk, M. Wahlström, Solving hard cut problems via flow-augmentation, in: D. Marx (Ed.), Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms, SODA 2021, Virtual Conference, January 10 - 13, 2021, SIAM, 2021, pp. 149–168.
[20] E.J. Kim, S. Kratsch, M. Pilipczuk, M. Wahlström, Directed flow-augmentation, in: S. Leonardi, A. Gupta (Eds.), STOC '22: 54th Annual ACM SIGACT Symposium on Theory of Computing, Rome, Italy, June 20 - 24, 2022, ACM, 2022, pp. 938–947.
[21] M. Cygan, F.V. Fomin, L. Kowalik, D. Lokshtanov, D. Marx, M. Pilipczuk, M. Pilipczuk, S. Saurabh, Parameterized Algorithms, Springer, 2015.
[22] J. Chen, Y. Liu, S. Lu, An improved parameterized algorithm for the minimum node multiway cut problem, Algorithmica 55 (1) (2009) 1–13, https://doi.org/10.1007/s00453-007-9130-6.
[23] D. Lokshtanov, D. Marx, Clustering with local restrictions, Inf. Comput. 222 (2013) 278–292, https://doi.org/10.1016/j.ic.2012.10.016.
[24] H.L. Bodlaender, A. Dawar, V.V. Williams, EATCS-IPEC Nerode Prize, 2020 (2020), https://eatcs.org/index.php/component/content/article/1-news/2861-eatcs-ipec-nerode-prize-2020-.
[25] D. Lokshtanov, F. Panolan, S. Saurabh, R. Sharma, M. Zehavi, Covering small independent sets and separators with applications to parameterized algorithms, ACM Trans. Algorithms 16 (3) (2020) 32:1–32:31, https://doi.org/10.1145/3379698.
[26] R. Bevern, T. Fluschnik, O.Y. Tsidulko, Parameterized algorithms and data reduction for the short secluded s-t-path problem, Networks 75 (1) (2020) 34–63, https://doi.org/10.1002/net.21904.
[27] S. Chechik, M.P. Johnson, M. Parter, D. Peleg, Secluded connectivity problems, Algorithmica 79 (3) (2017) 708–741, https://doi.org/10.1007/s00453-016-0222-z.
[28] M. Luckow, T. Fluschnik, On the computational complexity of length- and neighborhood-constrained path problems, Inf. Process. Lett. 156 (2020) 105913, https://doi.org/10.1016/j.ipl.2019.105913.
[29] H. Donkers, B.M.P. Jansen, J.J.H. de Kroon, Finding k-secluded trees faster, J. Comput. Syst. Sci. 138 (2023) 103461, https://doi.org/10.1016/J.JCSS.2023.05.006.
[30] F.V. Fomin, P.A. Golovach, N. Karpov, A.S. Kulikov, Parameterized complexity of secluded connectivity problems, Theory Comput. Syst. 61 (3) (2017) 795–819, https://doi.org/10.1007/s00224-016-9717-x.
[31] R. Bevern, T. Fluschnik, G.B. Mertzios, H. Molter, M. Sorge, O. Suchý, The parameterized complexity of finding secluded solutions to some classical optimization problems on graphs, Discrete Optim. 30 (2018) 20–50, https://doi.org/10.1016/j.disopt.2018.05.002.
[32] P.A. Golovach, P. Heggernes, P.T. Lima, P. Montealegre, Finding connected secluded subgraphs, J. Comput. Syst. Sci. 113 (2020) 101–124, https://doi.org/10.1016/j.jcss.2020.05.006.
[33] R. Ganian, M.S. Ramanujan, S. Szeider, Discovering archipelagos of tractability for constraint satisfaction and counting, ACM Trans. Algorithms 13 (2) (2017) 29:1–29:32, https://doi.org/10.1145/3014587.
[34] A. Jacob, J.J.H. de Kroon, D. Majumdar, V. Raman, Deletion to scattered graph classes I - case of finite number of graph classes, J. Comput. Syst. Sci. 138 (2023) 103460, https://doi.org/10.1016/j.jcss.2023.05.005.
[35] A. Jacob, D. Majumdar, V. Raman, Deletion to scattered graph classes II - improved FPT algorithms for deletion to pairs of graph classes, J. Comput. Syst. Sci. 136 (2023) 280–301, https://doi.org/10.1016/j.jcss.2023.03.004.
[36] A. Jacob, D. Majumdar, V. Raman, Parameterized complexity of deletion to scattered graph classes, in: Y. Cao, M. Pilipczuk (Eds.), 15th International Symposium on Parameterized and Exact Computation, IPEC 2020, December 14-18, 2020, Hong Kong, China (Virtual Conference), vol. 180 of LIPIcs, Schloss Dagstuhl - Leibniz-Zentrum Für Informatik, 2020, pp. 18:1–18:17, https://doi.org/10.4230/LIPIcs.IPEC.2020.18.
[37] A. Schrijver, Combinatorial Optimization - Polyhedra and Efficiency, Springer, 2003.
[38] S. Kratsch, M. Wahlström, Representative sets and irrelevant vertices: new tools for kernelization, J. ACM 67 (3) (2020) 16:1–16:50, https://doi.org/10.1145/3390887.
[39] M. Pilipczuk, M. Ziobro, Experimental evaluation of parameterized algorithms for graph separation problems: half-integral relaxations and matroid-based kernelization, CoRR, arXiv:1811.07779 [abs], 2018.
[40] M. Xiao, Simple and improved parameterized algorithms for multiterminal cuts, Theory Comput. Syst. 46 (4) (2010) 723–736, https://doi.org/10.1007/s00224-009-9215-5.