Finding k-Secluded Trees Faster

Huib Donkers, Bart M. P. Jansen(✉), and Jari J. H. de Kroon

Eindhoven University of Technology, Eindhoven, The Netherlands
{h.t.donkers,b.m.p.jansen,j.j.h.d.kroon}@tue.nl

Abstract. We revisit the k-Secluded Tree problem. Given a vertex-weighted undirected graph G, its objective is to find a maximum-weight induced subtree T whose open neighborhood has size at most k. We present a fixed-parameter tractable algorithm that solves the problem in time $2^{\mathcal{O}(k \log k)} \cdot n^{\mathcal{O}(1)}$, improving on a double-exponential running time from earlier work by Golovach, Heggernes, Lima, and Montealegre. Starting from a single vertex, our algorithm grows a k-secluded tree by branching on vertices in the open neighborhood of the current tree T. To bound the branching depth, we prove a structural result that can be used to identify a vertex that belongs to the neighborhood of any k-secluded supertree $T' \supseteq T$ once the open neighborhood of T becomes sufficiently large. We extend the algorithm to enumerate compact descriptions of all maximum-weight k-secluded trees, which allows us to count the number of such trees containing a specified vertex in the same running time.

Keywords: Secluded tree · FPT · Enumeration algorithm

1 Introduction

Background. We revisit a problem from the field of parameterized complexity: Given a graph G with positive weights on the vertices, find a connected induced acyclic subgraph H of maximum weight such that the open neighborhood of H in G has size at most k.

A parameterized problem is fixed parameter tractable (FPT) [4,6] if there is an algorithm that, given an instance I with parameter k, solves the problem in time $f(k) \cdot |I|^{\mathcal{O}(1)}$ for some computable function f. For problems that are FPT, such algorithms allow NP-hard problems to be solved efficiently on instances whose parameter is small. It is therefore desirable for the function f to grow slowly in terms of k, both out of theoretical interest as well as improving the practical relevance of these algorithms.

We say that a vertex set $S \subseteq V(G)$ is k-secluded in G if the open neighborhood of S in G has size at most k. An induced subgraph H of G is k-secluded in G if $V(H)$ is. If H is also a tree, we say that H is a k-secluded tree in G. Formally, the problem we study in this work is defined as follows.

B. M. P. Jansen—Supported by NWO Gravitation grant "Networks".
J. J. H. de Kroon—Supported by ERC Starting grant 803421, "ReduceSearch".

M. A. Bekos and M. Kaufmann (Eds.): WG 2022, LNCS 13453, pp. 173–186, 2022.
https://doi.org/10.1007/978-3-031-15914-5_13

Large Secluded Tree (LST)	**Parameter:** k
Input: An undirected graph G, a non-negative integer k, and a weight function $w\colon V(G) \to \mathbb{N}^+$.	
Task: Find a k-secluded tree H of G of maximum weight, or report that no such H exists.	

Golovach et al. [11] consider the more general Connected Secluded Π-Subgraph, where the k-secluded induced subgraph of G should belong to some target graph class Π. They mention that (Large) Secluded Tree is FPT and can be solved in time $2^{2^{\mathcal{O}(k \log k)}} \cdot n^{\mathcal{O}(1)}$ using the recursive understanding technique, the details of which can be found in the arXiv version [10]. For the case where Π is characterized by a finite set of forbidden induced subgraphs $\mathcal{F}$, they show that the problem is FPT with a triple-exponential dependency. They pose the question whether it is possible to avoid these double- and triple-exponential dependencies on the parameter. They give some examples of Π for which this is the case, namely for Π being a clique, a star, a d-regular graph, or an induced path.

Results. Our main result is an algorithm for Large Secluded Tree that takes $2^{\mathcal{O}(k \log k)} \cdot n^4$ time. This answers the question of Golovach et al. [11] affirmatively for the case of trees. We solve a more general version of the problem, where a set of vertices is given that should be part of the k-secluded tree. Our algorithm goes one step further by allowing us to find all maximum-weight solutions. As we will later argue, it is not possible to output all such solutions directly in the promised running time. Instead, the output consists of a bounded number of solution descriptions such that each maximum-weight solution can be constructed from one such description. This is similar in spirit to the work of Guo et al. [12], who enumerate all minimal solutions to the Feedback Vertex Set problem in $\mathcal{O}(c^k \cdot m)$ time. They do so by giving a list of *compact representations*, a set $\mathcal{C}$ of pairwise disjoint vertex subsets such that choosing exactly one vertex from every set results in a minimal feedback vertex set. Our descriptions are *non-redundant* (no two descriptions describe the same secluded tree), which allows us to *count* the number of maximum-weight k-secluded trees containing a specified vertex in the same running time.

Techniques. Rather than using recursive understanding, our algorithm is based on bounded-depth branching with a non-trivial progress measure. Similarly to existing algorithms to compute spanning trees with many leaves [13], our algorithm iteratively grows the vertex set of a k-secluded tree T. If we select a vertex v in the neighborhood of the current tree T, then for any k-secluded supertree T' of T there are two possibilities: either v belongs to the neighborhood of T', or it is contained in T'; the latter case can only happen if v has exactly one neighbor in T. Solutions of the first kind can be found by deleting v from the graph and searching for a $(k-1)$-secluded supertree of T. To find solutions of the second kind we can include v in T, but since the parameter does not decrease in this case we have to be careful that the recursion depth stays bounded. Using a

reduction rule to deal with degree-1 vertices, we can essentially ensure that v has at least three neighbors (exactly one of which belongs to T), so that adding v to T strictly increases the open neighborhood size $|N(T)|$. Our main insight to obtain an FPT algorithm is a structural lemma showing that, whenever $|N(T)|$ becomes sufficiently large in terms of k, we can identify a vertex u that belongs to the open neighborhood of any k-secluded supertree $T' \supseteq T$. At that point, we can remove u and decrease k to make progress.

Related Work. Secluded versions of several classic optimization problems have been studied intensively in recent years [1–3,8,14], many of which are discussed in Till Fluschnik's PhD thesis [7]. Marx [15] considers a related problem Cutting k (connected) vertices, where the aim is to find a (connected) set S of size exactly k with at most ℓ neighbors. Without the connectivity requirement, the problem is W[1]-hard by $k + \ell$. The problem becomes FPT when S is required to be connected, but remains W[1]-hard by k and ℓ seperately. Fomin et al. [9] consider the variant where $|S| \leq k$ and show that it is FPT parameterized by ℓ.

Organization. We introduce our enumeration framework in Sect. 2. We present our algorithm that enumerates maximum-weight k-secluded trees in Sect. 3 and present its correctness and running time analyses. We give some conclusions in Sect. 4. Proofs marked with ★ are deferred to the full version [5].

2 Framework for Enumerating Secluded Trees

We consider simple undirected graphs with vertex set $V(G)$ and edge set $E(G)$. We use standard notation pertaining to graph algorithms, such as presented by Cygan et al. [4]. When the graph G is clear from context, we denote $|V(G)|$ and $|E(G)|$ by n and m respectively. The open neighborhood of a vertex set X in a graph G is denoted by $N_G(X)$, where the subscript may be omitted if G is clear from context. For a subgraph H of G we may write $N(H)$ to denote $N(V(H))$. If $w\colon V(G) \to \mathbb{N}^+$ is a weight function, then for any $S \subseteq V(G)$ let $w(S) := \sum_{v \in S} w(s)$ and for any subgraph H of G we may denote $w(V(H))$ by $w(H)$.

It is not possible to enumerate all maximum-weight k-secluded trees in FPT time; consider the graph with n vertices of weight 1 and two vertices of weight n which are connected by $k+1$ vertex-disjoint paths on $n/(k+1)$ vertices each, then there are $\mathcal{O}(k \cdot (n/k)^k)$ maximum-weight k-secluded trees which consist of all vertices except one vertex out of exactly k paths. However, it is possible to give one short description for such an exponential number of k-secluded trees.

Definition 1. *For a graph G, a* description *is a pair $(r, \mathcal{X})$ consisting of a vertex $r \in V(G)$ and a set $\mathcal{X}$ of pairwise disjoint subsets of $V(G - r)$ such that for any set S consisting of exactly one vertex from each set $X \in \mathcal{X}$, the connected component H of $G - S$ containing r is acyclic and $N(H) = S$, i.e., H is a $|\mathcal{X}|$-secluded tree in G. The* order *of a description is equal to $|\mathcal{X}|$. We say that a k-secluded tree H is* described by *a description $(r, \mathcal{X})$ if $N(H)$ consists of exactly one vertex of each $X \in \mathcal{X}$ and $r \in V(H)$.*

Note that a single k-secluded tree H can be described by multiple descriptions. For example, for a path on $v_1, \ldots, v_4$ the 1-secluded tree induced by $\{v_1, v_2\}$ is described by $(v_1, \{\{v_3, v_4\}\})$, $(v_1, \{\{v_3\}\})$, and $(v_2, \{\{v_3\}\})$. We define the concept of redundancy in a set of descriptions.

Definition 2. *For a graph G, a set of descriptions $\mathfrak{X}$ of maximum order k is called* redundant *for G if there is a k-secluded tree H in G such that H is described by two distinct descriptions in $\mathfrak{X}$. We say $\mathfrak{X}$ is* non-redundant *for G otherwise.*

Definition 3. *For a graph G and a set of descriptions $\mathfrak{X}$ of maximum order k, let $\mathcal{T}_G(\mathfrak{X})$ denote the set of all k-secluded trees in G described by a description in $\mathfrak{X}$.*

Note 1. For a graph G we have $\mathcal{T}_G(\mathfrak{X}_1) \cup \mathcal{T}_G(\mathfrak{X}_2) = \mathcal{T}_G(\mathfrak{X}_1 \cup \mathfrak{X}_2)$ for any two sets of descriptions $\mathfrak{X}_1, \mathfrak{X}_2$.

Note 2. For a graph G, a set of descriptions $\mathfrak{X}$, and non-empty vertex sets X_1, X_2 disjoint from $\bigcup_{(r,\mathcal{X})\in\mathfrak{X}}(\{r\} \cup \bigcup_{X\in\mathcal{X}} X)$, the set $\mathcal{T}_G(\{(r, \mathcal{X} \cup \{X_1 \cup X_2\}) \mid (r, \mathcal{X}) \in \mathfrak{X})\}$ equals $\mathcal{T}_G(\{(r, \mathcal{X} \cup \{X_1\}) \mid (r, \mathcal{X}) \in \mathfrak{X})\} \cup (r, \mathcal{X} \cup \{X_2\}) \mid (r, \mathcal{X}) \in \mathfrak{X})\})$.

For an induced subgraph H of G and a set $F \subseteq V(G)$, we say that H is a supertree of F if H induces a tree and $F \subseteq V(H)$. Let $\mathcal{S}_G^k(F)$ be the set of all k-secluded supertrees of F in G. For a set X of subgraphs of G let $\text{maxset}_w(X) := \{H \in X \mid w(H) \geq w(H') \text{ for all } H' \in X\}$. We focus our attention to the following version of the problem, where some parts of the tree are already given.

Enumerate Large Secluded Supertrees (ELSS) **Parameter:** k
Input: A graph G, a non-negative integer k, non-empty vertex sets $T \subseteq F \subseteq V(G)$ such that $G[T]$ is connected, and a weight function $w\colon V(G) \to \mathbb{N}^+$.
Output: A non-redundant set $\mathfrak{X}$ of descriptions such that $\mathcal{T}_G(\mathfrak{X}) = \text{maxset}_w(\mathcal{S}_G^k(F))$.

Note that if $G[T]$, or even $G[F]$, contains a cycle, then the answer is trivially the empty set. In the end we solve the general enumeration problem by solving ELSS with $F = T = \{v\}$ for each $v \in V(G)$ and reporting only those k-secluded trees of maximum weight. Intuitively, our algorithm for ELSS finds k-secluded trees that "grow" out of T. In order to derive some properties of the types of descriptions we compute, we may at certain points demand that certain vertices non-adjacent to T need to end up in the k-secluded tree. For this reason the input additionally has a set F, rather than just T.

Our algorithm solves smaller instances recursively. We use the following abuse of notation: in an instance with graph G and weight function $w\colon V(G) \to \mathbb{N}^+$, when solving the problem recursively for an instance with induced subgraph G' of G, we keep the weight function w instead of restricting the domain of w to $V(G')$.

Note 3. For a graph G, a vertex $v \in V(G)$, and an integer $k \geq 1$, if H is a $(k-1)$-secluded tree in $G-v$, then H is a k-secluded tree in G. Consequently, $\mathcal{S}^{k-1}_{G-v}(F) \subseteq \mathcal{S}^{k}_{G}(F)$ for any $F \subseteq V(G)$.

Note 4. For a graph G, a vertex $v \in V(G)$, and an integer $k \geq 1$, if H is a k-secluded tree in G with $v \in N_G(H)$, then H is a $(k-1)$-secluded tree in $G-v$. Consequently, $\{H \in \mathcal{S}^{k}_{G}(F) \mid v \in N_G(H)\} \subseteq \mathcal{S}^{k-1}_{G-v}(F)$ for any $F \subseteq V(G)$.

3 Enumerate Large Secluded Supertrees

Section 3.1 proves the correctness of a few subroutines used by the algorithm. Section 3.2 describes the algorithm to solve ELSS. In Sect. 3.3 we prove its correctness and in Sect. 3.4 we analyze its time complexity. In Sect. 3.5 we show how the algorithm for ELSS can be used to count and enumerate maximum-weight k-secluded trees containing a specified vertex.

3.1 Subroutines for the Algorithm

Similar to the Feedback Vertex Set algorithm given by Guo et al. [12], we aim to get rid of degree-1 vertices. In our setting there is one edge case however. The reduction rule is formalized as follows.

Reduction Rule 1. *For an* ELSS *instance* (G, k, F, T, w) *with a degree-1 vertex* v *in* G *such that* $F \neq \{v\}$, contracting v into *its neighbor* u *yields the* ELSS *instance* $(G-v, k, F', T', w')$ *where the weight of* u *is increased by* $w(v)$ *and:*

$$F' = \begin{cases} (F \backslash \{v\}) \cup \{u\} & \text{if } v \in F \\ F & \text{otherwise} \end{cases} \qquad T' = \begin{cases} (T \backslash \{v\}) \cup \{u\} & \text{if } v \in T \\ T & \text{otherwise.} \end{cases}$$

We prove the correctness of the reduction rule, that is, the descriptions of the reduced instance form the desired output for the original instance.

Lemma 1 (★). *Let* $I = (G, k, F, T, w)$ *be an* ELSS *instance. Suppose* G *contains a degree-1 vertex* v *such that* $\{v\} \neq F$. *Let* $I' = (G-v, k, F', T', w')$ *be the instance obtained by contracting* v *into its neighbor* u. *If* $\mathfrak{X}$ *is a non-redundant set of descriptions for* $G-v$ *such that* $\mathcal{T}_{G-v}(\mathfrak{X}) = maxset_{w'}(\mathcal{S}^{k}_{G-v}(F'))$, *then* $\mathfrak{X}$ *is a non-redundant set of descriptions for* G *such that* $\mathcal{T}_{G}(\mathfrak{X}) = maxset_{w}(\mathcal{S}^{k}_{G}(F))$.

We say an instance is *almost leafless* if the lemma above cannot be applied, that is, if G contains a vertex v of degree 1, then $F = \{v\}$.

Lemma 2. *There is an algorithm that, given an almost leafless* ELSS *instance* (G, k, F, T, w) *such that* $k > 0$ *and* $|N_G(T)| > k(k+1)$, *runs in time* $\mathcal{O}(k \cdot n^3)$ *and either:*

1. *finds a vertex* $v \in V(G) \backslash F$ *such that any* k*-secluded supertree* H *of* F *in* G *satisfies* $v \in N_G(H)$, *or*
2. *concludes that* G *does not contain a* k*-secluded supertree of* F.

Proof. We aim to find a vertex $v \in V(G) \backslash F$ with $k+2$ distinct paths $P_1, \ldots, P_{k+2}$ from $N_G(T)$ to v that intersect only in v and do not contain vertices from T. We first argue that such a vertex v satisfies the first condition, if it exists. Consider some k-secluded supertree H of F. Since the paths $P_1, \ldots, P_{k+2}$ are disjoint apart from their common endpoint v while $|N_G(H)| \leq k$, there are two paths P_i, P_j with $i \neq j \in [k+2]$ for which $P_i \backslash \{v\}$ and $P_j \backslash \{v\}$ do not intersect $N_G(H)$. These paths are contained in H since they contain a neighbor of $T \subseteq F \subseteq H$. As P_i and P_j form a cycle together with a path through the connected set T, which cannot be contained in the acyclic graph H, this implies $v \in N_G(H)$.

Next we argue that if G has a k-secluded supertree H of $F \supseteq T$, then there exists such a vertex v. Consider an arbitrary such H and root it at a vertex $t \in T$. For each vertex $u \in N_G(T)$, we construct a path P_u disjoint from T that starts in u and ends in $N_G(H)$, as follows.

- If $u \notin H$, then $u \in N_G(H)$ and we take $P_u = (u)$.
- If $u \in H$, then let ℓ_u be an arbitrary leaf in the subtree of H rooted at u; possibly $u = \ell_u$. Since T is connected and $H \supseteq T$ is acyclic and rooted in $t \in T$, the subtree rooted at $u \in N_G(T) \cap H$ is disjoint from T. Hence $\ell_u \notin T$, so that $F \neq \{\ell_u\}$. As the instance is almost leafless we therefore have $\deg_G(\ell_u) > 1$. Because ℓ_u is a leaf of H this implies that $N_G(\ell_u)$ contains a vertex y other than the parent of ℓ_u in H, so that $y \in N_G(H)$. We let P_u be the path from u to ℓ_u through H, followed by the vertex $y \in N_G(H)$.

The paths we construct are distinct since their startpoints are. Two constructed paths cannot intersect in any vertex other than their endpoints, since they were extracted from different subtrees of H. Since we construct $|N_G(T)| > k(k+1)$ paths, each of which ends in $N_G(H)$ which has size at most k, some vertex $v \in N_G(H)$ is the endpoint of $k+2$ of the constructed paths. As shown in the beginning the proof, this establishes that v belongs to the neighborhood of any k-secluded supertree of F. Since $F \subseteq V(H)$ we have $v \notin F$.

All that is left to show is that we can find such a vertex v in the promised time bound. After contracting T into a source vertex s, for each $v \in V(G) \backslash F$, do $k+2$ iterations of the Ford-Fulkerson algorithm in order to check if there are $k+2$ internally vertex-disjoint sv-paths. If so, then return v. If for none of the choices of v this holds, then output that there is no k-secluded supertree of F in G. In order to see that this satisfies the claimed running time bound, note that there are $\mathcal{O}(n)$ choices for v, and $k+2$ iterations of Ford-Fulkerson runs can be implemented to run in $\mathcal{O}(k \cdot (n+m))$ time. □

3.2 The Algorithm

Consider an input instance (G, k, F, T, w) of ELSS. If $G[F]$ contains a cycle, return $\emptyset$. Otherwise we remove all connected components of G that do not contain a vertex of F. If more than one connected component remains, return $\emptyset$. Then, while there is a degree-1 vertex v such that $F \neq \{v\}$, contract v into its neighbor as per Rule 1. While $N_G(T)$ contains a vertex $v \in F$, add v to T. Finally, if $N_G(F) = \emptyset$, return $\{(r, \emptyset)\}$ for some $r \in F$. Otherwise if $k = 0$, return $\emptyset$.

We proceed by considering the neighborhood of T as follows:

1. If any vertex $v \in N_G(T)$ has two neighbors in T, then recursively run this algorithm to obtain a set of descriptions $\mathfrak{X}'$ for $(G - v, k - 1, F, T, w)$ and return $\{(r, \mathcal{X} \cup \{\{v\}\}) \mid (r, \mathcal{X}) \in \mathfrak{X}'\}$.
2. If $|N_G(T)| > k(k+1)$, apply Lemma 2. If it concludes that G does not contain a k-secluded supertree of F, return $\emptyset$. Otherwise let $v \in V(G) \backslash F$ be the vertex it finds, obtain a set of descriptions $\mathfrak{X}'$ for $(G - v, k - 1, F, T, w)$ and return $\{(r, \mathcal{X} \cup \{\{v\}\}) \mid (r, \mathcal{X}) \in \mathfrak{X}'\}$.
3. Pick some $v \in N_G(T)$ and let $P = (v = v_1, v_2, \ldots, v_\ell)$ be the unique[1] maximal path disjoint from T satisfying $\deg_G(v_i) = 2$ for each $1 \leq i < \ell$ and ($v_\ell \in N_G(T)$ or $\deg_G(v_\ell) > 2$).
   (a) If $v_\ell \notin F$, obtain a set of descriptions $\mathfrak{X}_1$ by recursively solving $(G - v_\ell, k - 1, F, T, w)$. Otherwise take $\mathfrak{X}_1 = \emptyset$. (We find the k-secluded trees avoiding v_ℓ but containing $P - v_\ell$.)
   (b) If $P - F - v_\ell \neq \emptyset$, obtain a set of descriptions $\mathfrak{X}_2$ by recursively solving $(G - V(P - v_\ell), k - 1, (F \backslash V(P)) \cup \{v_\ell\}, T, w)$. Otherwise take $\mathfrak{X}_2 = \emptyset$. (We find the k-secluded trees containing both endpoints of P which have one vertex in P as a neighbor.)
   (c) If $G[F \cup V(P)]$ is acyclic, obtain a set of descriptions $\mathfrak{X}_3$ by recursively solving $(G, k, F \cup V(P), T \cup V(P), w)$. Otherwise take $\mathfrak{X}_3 = \emptyset$. (We find the k-secluded trees containing the entire path P.)

   Let M be the set of minimum weight vertices in $P - F - v_\ell$ and define:

$$\mathfrak{X}'_1 := \{(r, \mathcal{X} \cup \{\{v_\ell\}\}) \mid (r, \mathcal{X}) \in \mathfrak{X}_1\}$$
$$\mathfrak{X}'_2 := \{(r, \mathcal{X} \cup \{M\}) \mid (r, \mathcal{X}) \in \mathfrak{X}_2\}$$
$$\mathfrak{X}'_3 := \mathfrak{X}_3.$$

   For each $i \in [3]$ let w_i be the weight of an arbitrary $H \in \mathcal{T}_G(\mathfrak{X}'_i)$, or 0 if $\mathfrak{X}'_i = \emptyset$. Return the set $\mathfrak{X}'$ defined as $\bigcup_{\{i \in [3] \mid w_i = \max\{w_1, w_2, w_3\}\}} \mathfrak{X}'_i$.

3.3 Proof of Correctness

In this section we argue that the algorithm described in Sect. 3.2 solves the ELSS problem. In various steps we identify a vertex v such that the neighborhood of any (maximum-weight) k-secluded supertree must include v. We argue that for these steps, the descriptions of the current instance can be found by adding $\{v\}$ to every description of the supertrees of T in $G - v$ if some preconditions are satisfied.

Lemma 3 (★). *Let (G, k, F, T, w) be an* ELSS *instance and let $v \in V(G) \backslash F$. Let $\mathfrak{X}$ be a set of descriptions for $G - v$ such that $\mathcal{T}_{G-v}(\mathfrak{X}) = maxset_w(\mathcal{S}^{k-1}_{G-v}(F))$ and $v \in N_G(H)$ for all $H \in \mathcal{T}_{G-v}(\mathfrak{X})$. Then we have:*

$$\mathcal{T}_G\left(\{(r, \mathcal{X} \cup \{\{v\}\}) \mid (r, \mathcal{X}) \in \mathfrak{X}\}\right) = maxset_w\{H \in \mathcal{S}^k_G(F) \mid v \in N_G(H)\}.$$

[1] To construct P, initialize $P := (v = v_1)$; then while $\deg_G(v_{|V(P)|}) = 2$ and $N_G(v_{|V(P)|}) \backslash (V(P) \cup T)$ consists of a single vertex, append that vertex to P.

The following lemma is used to argue that the branches of Step 3 are disjoint.

Lemma 4. *Let* (G, k, F, T, w) *be an almost leafless* ELSS *instance such that* G *is connected and* $N_G(F) \neq \emptyset$*. Fix some* $v \in N_G(T)$ *and let* $P = (v = v_1, v_2, \ldots, v_\ell)$ *be the unique maximal path disjoint from* T *satisfying* $deg_G(v_i) = 2$ *for each* $1 \leq i < \ell$ *and* ($v_\ell \in N_G(T)$ *or* $deg_G(v_\ell) > 2$*). Then for any* maximum-weight k*-secluded supertree* H *of* F*, exactly one of the following holds:*

1. $v_\ell \in N(H)$ *(so* $v_\ell \notin F$*),*
2. $|N(H) \cap V(P - F - v_\ell)| = 1$ *and* $v_\ell \in V(H)$*, or*
3. $V(P) \subseteq V(H)$*.*

Proof. First note that such a vertex v exists since $N_G(F) \neq \emptyset$ and G is connected, so $N_G(T) \neq \emptyset$. Furthermore since the instance is almost leafless, the path P is well defined. If there is no k-secluded supertree of F, then there is nothing to show. So suppose H is a maximum-weight k-secluded supertree of F. We have $v \in V(P)$ is a neighbor of $T \subseteq F \subseteq V(H)$, so either $V(P) \subseteq V(H)$ or $V(P)$ contains a vertex from $N(H)$. In the first case Item 3 holds, in the second case we have $|N(H) \cap V(P)| \geq 1$. First suppose that $|N(H) \cap V(P)| \geq 2$. Let $i \in [\ell]$ be the smallest index such that $v_i \in N(H) \cap V(P)$. Similarly let $j \in [\ell]$ be the largest such index. We show that in this case we can contradict the fact that H is a maximum-weight k-secluded supertree of F. Observe that $H' = V(H) \cup \{v_i, \ldots, v_{j-1}\}$ induces a tree since $(v_i, \ldots, v_{j-1})$ forms a path of degree-2 vertices and the neighbor v_j of v_{j-1} is not in H. Furthermore H' has a strictly smaller neighborhood than H and it has larger weight as vertices have positive weight. Since $F \subseteq V(H')$, this contradicts that H is a maximum-weight k-secluded supertree of F.

We conclude that $|N(H) \cap V(P)| = 1$. Let $i \in [\ell]$ be the unique index such that $N(H) \cap V(P) = \{v_i\}$. Clearly $v_i \notin F$. In the case that $i = \ell$, then Item 1 holds. Otherwise if $i < \ell$, the first condition of Item 2 holds. In order to argue that the second condition also holds, suppose that $v_\ell \notin V(H)$. Then $H \cup \{v_i, \ldots, v_{\ell-1}\}$ is a k-secluded supertree of F in G and it has larger weight than H as vertices have positive weight. This contradicts the fact that H has maximum weight, hence the second condition of Item 2 holds as well. □

Armed with Lemma 4 we are now ready to prove correctness of the algorithm.

Lemma 5. *The algorithm described in Sect. 3.2 is correct.*

Proof. Let $I = (G, k, F, T, w)$ be an ELSS instance. We prove correctness by induction on $|V(G) \backslash F|$. Assume the algorithm is correct for any input $(\hat{G}, \hat{k}, \hat{F}, \hat{T}, \hat{w})$ with $|V(\hat{G}) \backslash \hat{F}| < |V(G) \backslash F|$. We prove correctness of the algorithm up to Step 3. The correctness of Step 3 is proven in the full version [5].

Before Step 1. We first prove correctness when the algorithm terminates before Step 1, which includes the base case of the induction. Note that if $G[F]$ contains a cycle, then no induced subgraph H of G with $F \subseteq V(H)$ can be acyclic. Therefore the set of maximum-weight k-secluded trees containing F is the empty

set, so we correctly return $\emptyset$. Otherwise $G[F]$ is acyclic. Clearly any connected component of G that has no vertices of F can be removed. If there are two connected components of G containing vertices of F, then no induced subgraph of G containing all of F can be connected, again we correctly return the empty set. In the remainder we have that G is connected.

By iteratively applying Lemma 1 we conclude that a solution to the instance obtained after iteratively contracting (most) degree-1 vertices is also a solution to the original instance. Hence we can proceed to solve the new instance, which we know is almost leafless. In addition, observe that the contraction of degree-1 vertices maintains the property that G is connected and $G[F]$ is acyclic.

After exhaustively adding vertices $v \in N_G(T) \cap F$ to T we have that $G[T]$ is a connected component of $G[F]$. In the case that $N_G(F) = \emptyset$, then since G is connected it follows that $F = T = V(G)$ and therefore T is the only maximum-weight k-secluded tree. For any $r \in V(G)$, the description $(r, \emptyset)$ describes this k-secluded tree, so we return $\{(r, \emptyset)\}$. In the remainder we have $N_G(F) \neq \emptyset$.

Since $N_G(F) \neq \emptyset$ and G is almost leafless, we argue that there is no 0-secluded supertree of F. Suppose G contains a 0-secluded supertree H of F, so $|N_G(H)| = 0$ and since $H \supseteq F$ is non-empty and G is connected we must have $H = G$, hence G is a tree with at least two vertices (since F and $N_G(F)$ are both non-empty) so G contains at least two vertices of degree-1, contradicting that G is almost leafless. So there is no k-secluded supertree of F in G and the algorithm correctly returns $\emptyset$ if $k = 0$.

Observe that the value $|V(G) \setminus F|$ cannot have increased since the start of the algorithm since we never add vertices to G and any time we remove a vertex from F it is also removed from G. Hence we can still assume in the remainder of the proof that the algorithm is correct for any input $(\hat{G}, \hat{k}, \hat{F}, \hat{T}, \hat{w})$ with $|V(\hat{G}) \setminus \hat{F}| < |V(G) \setminus F|$. To conclude this part of the proof, we have established that if the algorithm terminates before reaching Step 1, then its output is correct. On the other hand, if the algorithm continues we can make use of the following properties of the instance just before reaching Step 1:

Property 1. If the algorithm does not terminate before reaching Step 1 then (i) the ELSS instance (G, k, F, T, w) is almost leafless, (ii) $G[F]$ is acyclic, (iii) $G[T]$ is a connected component of $G[F]$, (iv) G is connected, (v) $k > 0$, and (vi) $N_G(F) \neq \emptyset$.

Step 1. Before arguing that the return value in Step 1 is correct, we observe the following.

Claim 1. If H is an induced subtree of G that contains T and $v \in N_G(T)$ has at least two neighbors in T, then $v \in N_G(H)$.

Proof. Suppose $v \notin N_G(H)$, then since $v \in N_G(T)$ and $T \subseteq V(H)$ we have that $v \in V(H)$. But then since T is connected, subgraph H contains a cycle. This contradicts that H is a tree and confirms that $v \in N_G(H)$. ⌟

Now consider the case that in Step 1 we find a vertex $v \in N_G(T)$ with two neighbors in T, and let $\mathfrak{X}'$ be the set of descriptions as obtained by the algorithm

through recursively solving the instance $(G - v, k - 1, F, T, w)$. Since $|V(G - v)\backslash F| < |V(G)\backslash F|$ (as $v \notin F$) we know by induction that $\mathcal{T}_{G-v}(\mathfrak{X}')$ is the set of all maximum-weight $(k-1)$-secluded supertrees of F in $G-v$. Any $H \in \mathcal{T}_{G-v}(\mathfrak{X}')$ is an induced subtree of G with $T \subseteq V(H)$, so by Claim 1 we have $v \in N_G(H)$ for all $H \in \mathcal{T}_{G-v}(\mathfrak{X}')$. We can now apply Lemma 3 to conclude that $\mathcal{T}_G(\{(r, \mathcal{X} \cup \{\{v\}\}) \mid (r, \mathcal{X}) \in \mathfrak{X}'\})$ is the set of all maximum-weight k-secluded supertrees H of F in G for which $v \in N_G(H)$. Again by Claim 1 we have that $v \in N_G(H)$ for all such k-secluded supertrees of F, hence $\mathcal{T}_G(\{(r, \mathcal{X} \cup \{\{v\}\}) \mid (r, \mathcal{X}) \in \mathfrak{X}'\})$ is the set of all maximum-weight k-secluded supertrees of F in G. We argue non-redundancy of the output. Suppose that two descriptions $(r, \mathcal{X} \cup \{\{v\}\})$ and $(r', \mathcal{X}' \cup \{\{v\}\})$ describe the same supertree H of F in G. Note that then $(r, \mathcal{X})$ and $(r', \mathcal{X})$ describe the same supertree H of F in $G - v$, which contradicts the induction hypothesis that the output of the recursive call was correct and therefore non-redundant.

Concluding this part of the proof, we showed that if the algorithm terminates during Step 1, then its output is correct. On the other hand, if the algorithm continues after Step 1 we can make use of the following in addition to Property 1.

Property 2. If the algorithm does not terminate before reaching Step 2 then no vertex $v \in N_G(T)$ has two neighbors in T.

Step 2. In Step 2 we use Lemma 2 if $|N_G(T)| > k(k+1)$. The preconditions of the lemma are satisfied since $k > 0$ and the instance is almost leafless by Property 1. If it concludes that G does not contain a k-secluded supertree of F, then the algorithm correctly outputs $\emptyset$. Otherwise it finds a vertex $v \in V(G)\backslash F$ such that any k-secluded supertree H of F in G satisfies $v \in N_G(H)$. We argue that the algorithm's output is correct. Let $\mathfrak{X}'$ be the set of descriptions as obtained through recursively solving $(G - v, k - 1, F, T, w)$. Since $v \notin F$ we have $|(V(G - v)\backslash F| < |V(G)\backslash F|$, so by induction we have that $\mathcal{T}_{G-v}(\mathfrak{X}')$ is the set of all maximum-weight $(k-1)$-secluded supertrees of F in $G - v$. Furthermore by Note 3 for any $H \in \mathcal{T}_{G-v}(\mathfrak{X}') = \mathcal{S}^{k-1}_{G-v}(F)$ we have $H \in \mathcal{S}^k_G(F)$, and therefore $v \in N_G(H)$. It follows that Lemma 3 applies to $\mathfrak{X}'$ so we can conclude that $\mathcal{T}_G(\{(r, \mathcal{X} \cup \{\{v\}\}) \mid (r, \mathcal{X}) \in \mathfrak{X}'\})$ is the set of maximum-weight k-secluded supertrees H of F in G for which $v \in N_G(H)$. Since we know there are no k-secluded supertrees H of F in G for which $v \notin N_G(H)$, it follows that $\mathcal{T}_G(\{(r, \mathcal{X} \cup \{\{v\}\}) \mid (r, \mathcal{X}) \in \mathfrak{X}\})$ is the set of maximum-weight k-secluded supertrees of F in G as required. Non-redundancy of the output follows as in Step 1.

To summarize the progress so far, we have shown that if the algorithm terminates before it reaches Step 3, then its output is correct. Alternatively, if we proceed to Step 3 we can make use of the following property, in addition to Properties 1 and 2, which we will use later in the running time analysis.

Property 3. If the algorithm does not terminate before reaching Step 3, then $|N_G(T)| \leq k(k+1)$.

Step 3 (★). In the full version [5] we show using Properties 1 to 3 that if the algorithm reaches Step 3, then its output is correct. For this we use Lemma 4 to argue that the k-secluded supertrees of F in G can be partitioned into three sets $\mathcal{T}_1, \mathcal{T}_2, \mathcal{T}_3$. The three recursive calls in Step 3 correspond to the subproblems of finding the secluded trees in $\mathcal{T}_1$, $\mathcal{T}_2$, and $\mathcal{T}_3$. Each call finds maximum-weight k-secluded trees of one particular type. Since the latter restriction may cause the tree to have smaller weight than maximum k-secluded trees in general, the postprocessing step of the algorithm restricts the output to describe only those types providing the maximum global weight. □

3.4 Runtime Analysis

If all recursive calls in the algorithm would decrease k then, since for $k = 0$ it does not make any further recursive calls, the maximum recursion depth is k. However in Step 3(c) the recursive call does not decrease k. In order to bound the recursion depth, we show the algorithm cannot make more than $k(k + 1)$ consecutive recursive calls in Step 3(c), that is, the recursion depth cannot increase by more than $k(k+1)$ since the last time k decreased. This follows from Lemma 6 together with the fact that if $N_G(T) > k(k + 1)$ then the algorithm executes Step 2, decreases k when it goes into recursion, and does not proceed to Step 3.

Lemma 6 (★). *If the recursion tree generated by the algorithm contains a path of $i \geq 1$ consecutive recursive calls in Step 3(c), and (G, k, F, T, w) is the instance considered in Step 3 where the i-th of these recursive calls is made, then $|N_G(T)| \geq i$.*

Using this bound on the number of consecutive recursive calls in Step 3(c), we obtain a maximum recursion depth of $\mathcal{O}(k^3)$. We argue that each recursive call takes $\mathcal{O}(kn^3)$ time and since we branch at most three ways, we obtain a running time of $3^{\mathcal{O}(k^3)} \cdot kn^3 = 3^{\mathcal{O}(k^3)} \cdot n^3$. However, with a more careful analysis we can give a better bound on the number of nodes in the recursion tree. For this, label each edge in the recursion tree with a label from the set $\{1, 2, 3a, 3b, 3c\}$ indicating where in the algorithm the recursive call took place. Now observe that each node in the recursion tree can be uniquely identified by a sequence of edge-labels corresponding to the path from the root of the tree to the relevant node. We call such a sequence of labels a *trace*. To bound the number of nodes in the recursion tree we give a bound on the number of valid traces. Since recursive calls corresponding to labels 1, 2, 3a, and 3b each decrease k, they can occur at most k times in a valid trace. All remaining labels in the trace are 3c. So the total number of traces of length ℓ is $\binom{\ell}{k} \cdot 4^k \leq \ell^k \cdot 4^k = (4\ell)^k$. Considering valid traces have a length of at most $k^2(k + 1)$ we derive the following bound on the total number of valid traces using the fact that $(k^c)^k = (2^{\log(k^c)})^k = 2^{\mathcal{O}(k \log k)}$:

$$\sum_{1 \leq \ell \leq k^2(k+1)} (4\ell)^k \leq k^2(k + 1) \cdot (4k^2(k + 1))^k = 2^{\mathcal{O}(k \log k)}.$$

We can conclude that the total number of nodes in the recursion tree is at most $2^{\mathcal{O}(k \log k)}$ which leads to the following lemma.

Lemma 7 (★). *The algorithm described in Sect. 3.2 can be implemented to run in time* $2^{\mathcal{O}(k \log k)} \cdot n^3$.

3.5 Finding, Enumerating, and Counting Large Secluded Trees

With the algorithm of Sect. 3.2 at hand we argue that we are able to enumerate k-secluded trees, count such trees containing a specified vertex, and solve LST.

Theorem 1 (★). *There is an algorithm that, given a graph G, weight function w, and integer k, runs in time $2^{\mathcal{O}(k \log k)} \cdot n^4$ and outputs a set of descriptions $\mathfrak{X}$ such that $\mathcal{T}_G(\mathfrak{X})$ is exactly the set of maximum-weight k-secluded trees in G. Each such tree H is described by $|V(H)|$ distinct descriptions in $\mathfrak{X}$.*

By returning an arbitrary maximum-weight k-secluded tree described by any description in the output of Theorem 1, we have the following consequence.

Corollary 1. *There is an algorithm that, given a graph G, weight function w, and integer k, runs in time $2^{\mathcal{O}(k \log k)} \cdot n^4$ and outputs a maximum-weight k-secluded tree in G if one exists.*

The following theorem captures the consequences for counting.

Theorem 2 (★). *There is an algorithm that, given a graph G, vertex $v \in V(G)$, weight function w, and integer k, runs in time $2^{\mathcal{O}(k \log k)} \cdot n^3$ and counts the number of k-secluded trees in G that contain v and have maximum weight out of all k-secluded trees containing v.*

4 Conclusion

We revisited the k-Secluded Tree problem first studied by Golovach et al. [11], leading to improved FPT algorithms with the additional ability to count and enumerate solutions. The non-trivial progress measure of our branching algorithm is based on a structural insight that allows a vertex that belongs to the *neighborhood* of every solution subtree to be identified, once the solution under construction has a sufficiently large open neighborhood. As stated, the correctness of this step crucially relies on the requirement that solution subgraphs are acyclic. It would be interesting to determine whether similar branching strategies can be developed to solve the more general k-Secluded Connected $\mathcal{F}$-Minor-Free Subgraph problem; the setting studied here corresponds to $\mathcal{F} = \{K_3\}$. While any $\mathcal{F}$-minor-free graph is known to be sparse, it may still contain large numbers of internally vertex-disjoint paths between specific pairs of vertices, which stands in the way of a direct extension of our techniques.

A second open problem concerns the optimal parameter dependence for k-Secluded Tree. The parameter dependence of our algorithm is $2^{\mathcal{O}(k \log k)}$. Can it be improved to single-exponential, or shown to be optimal under the Exponential Time Hypothesis?

References

1. van Bevern, R., Fluschnik, T., Mertzios, G.B., Molter, H., Sorge, M., Suchý, O.: The parameterized complexity of finding secluded solutions to some classical optimization problems on graphs. Discret. Optim. **30**, 20–50 (2018). https://doi.org/10.1016/j.disopt.2018.05.002
2. van Bevern, R., Fluschnik, T., Tsidulko, Y.O.: Parameterized algorithms and data reduction for the short secluded s-t-path problem. Networks **75**(1), 34–63 (2020). https://doi.org/10.1002/net.21904
3. Chechik, S., Johnson, M.P., Parter, M., Peleg, D.: Secluded connectivity problems. Algorithmica **79**(3), 708–741 (2016). https://doi.org/10.1007/s00453-016-0222-z
4. Cygan, M., et al.: Parameterized Algorithms. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-21275-3
5. Donkers, H., Jansen, B.M.P., de Kroon, J.J.H.: Finding k-secluded trees faster (2022). https://doi.org/10.48550/ARXIV.2206.09884
6. Downey, R.G., Fellows, M.R.: Parameterized Complexity. Monographs in Computer Science. Springer, Heidelberg (1999). https://doi.org/10.1007/978-1-4612-0515-9
7. Fluschnik, T.: Elements of efficient data reduction: fractals, diminishers, weights and neighborhoods. Ph.D. thesis, Technische Universität Berlin (2020). https://doi.org/10.14279/depositonce-10134
8. Fomin, F.V., Golovach, P.A., Karpov, N., Kulikov, A.S.: Parameterized complexity of secluded connectivity problems. Theory Comput. Syst. **61**(3), 795–819 (2016). https://doi.org/10.1007/s00224-016-9717-x
9. Fomin, F.V., Golovach, P.A., Korhonen, J.H.: On the parameterized complexity of cutting a few vertices from a graph. In: Chatterjee, K., Sgall, J. (eds.) MFCS 2013. LNCS, vol. 8087, pp. 421–432. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40313-2_38
10. Golovach, P.A., Heggernes, P., Lima, P.T., Montealegre, P.: Finding connected secluded subgraphs. CoRR, abs/1710.10979 (2017). arXiv:1710.10979
11. Golovach, P.A., Heggernes, P., Lima, P.T., Montealegre, P.: Finding connected secluded subgraphs. J. Comput. Syst. Sci. **113**, 101–124 (2020). https://doi.org/10.1016/j.jcss.2020.05.006
12. Guo, J., Gramm, J., Hüffner, F., Niedermeier, R., Wernicke, S.: Compression-based fixed-parameter algorithms for feedback vertex set and edge bipartization. J. Comput. Syst. Sci. **72**(8), 1386–1396 (2006). https://doi.org/10.1016/j.jcss.2006.02.001
13. Kneis, J., Langer, A., Rossmanith, P.: A new algorithm for finding trees with many leaves. Algorithmica **61**(4), 882–897 (2010). https://doi.org/10.1007/s00453-010-9454-5
14. Luckow, M.-J., Fluschnik, T.: On the computational complexity of length - and neighborhood-constrained path problems. Inf. Process. Lett. **156**, 105913 (2020). https://doi.org/10.1016/j.ipl.2019.105913
15. Marx, D.: Parameterized graph separation problems. Theor. Comput. Sci. **351**(3), 394–406 (2006). https://doi.org/10.1016/j.tcs.2005.10.007