Parameterized algorithms and data reduction for the short secluded $s$-$t$-path problem

René van Bevern
Department of Mechanics and Mathematics, Novosibirsk State University, Novosibirsk, Russian Federation, rvb@nsu.ru
Sobolev Institute of Mathematics, Siberian Branch of the Russian Academy of Sciences, Novosibirsk, Russian Federation

Till Fluschnik
Algorithmics and Computational Complexity, Fakultät IV, TU Berlin, Berlin, Germany
till.fluschnik@tu-berlin.de

Oxana Yu. Tsidulko
Sobolev Institute of Mathematics, Siberian Branch of the Russian Academy of Sciences, Novosibirsk, Russian Federation,
tsidulko@math.nsc.ru
Department of Mechanics and Mathematics, Novosibirsk State University, Novosibirsk, Russian Federation

Abstract
Given a graph $G = (V,E)$, two vertices $s, t \in V$, and two integers $k, \ell$, we search for a simple $s$-$t$-path with at most $k$ vertices and at most $\ell$ neighbors. For graphs with constant crossing number, we provide a subexponential $2^{O(\sqrt{n})}$-time algorithm, prove a matching lower bound, and show a polynomial-time data reduction algorithm that reduces any problem instance to an equivalent instance (a so-called problem kernel) of size polynomial in the vertex cover number of the input graph. In contrast, we show that the problem in general graphs is hard to preprocess. We obtain a $2^{O(tw)} \cdot \ell^2 \cdot n$-time algorithm for graphs of treewidth $tw$, show that there is no problem kernel with size polynomial in $tw$, yet show a problem kernels with size polynomial in the feedback edge number of the input graph and with size polynomial in the feedback vertex number, $k$, and $\ell$.

Keywords: NP-hard problem · fixed-parameter tractability · problem kernelization

1 Introduction
Finding shortest paths is a fundamental problem in route planning and has extensively been studied with respect to efficient algorithms, including data reduction and preprocessing [2]. In this work, we study the following NP-hard variant of finding shortest $s$-$t$-paths.

Problem 1.1 (Short Secluded Path (SSP)).
Instance: An undirected, simple graph $G = (V,E)$ with two distinct vertices $s, t \in V$, and two integers $k \geq 2$ and $\ell \geq 0$.
Question: Is there an $s$-$t$-path $P$ in $G$ such that $|V(P)| \leq k$ and $|N(V(P))| \leq \ell$?

The problem can be understood as finding short and safe routes for a convoy through a transportation network: each neighbor of the convoy’s travel path requires additional precaution. Thus, we seek to minimize not only the length of the convoy’s travel path, but also its number of neighbors. In our work, we study the above basic, unweighted variant, as well as weighted variants of the problem.

*A preliminary version of this work appeared in the Proceedings of the 18th Workshop on Algorithmic Approaches for Transportation Modeling, Optimization, and Systems (ATMOS 2018), 23–24 August, 2018, Helsinki, Finland [3]. This version contains full proof details, new kernelization results with respect to the feedback vertex number as parameter, and the algorithm for graphs of bounded treewidth has been generalized to a more general problem variant.
Table 1.1: Overview of our results. Herein, \(n\), tw, vc, fes, fvs, cr, and \(\Delta\) denote the number of vertices, treewidth, vertex cover number, feedback edge number, feedback vertex number, the crossing number, and maximum degree of the input graph, respectively.

|                        | On almost planar graphs (Section 2) | On tree-like graphs (Section 3) |
|------------------------|--------------------------------------|---------------------------------|
| exact solution         | \(2^{O(\sqrt{n})}\) time for constant cr (Theorem 2.1) | \(2^{O(\text{tw}) \cdot \ell^2 \cdot n}\) time (Theorem 3.2) |
| problem kernel         | size vc\((\sqrt{n})\) in \(K_{\ell,r}\)-free graphs (Theorem 2.5) | \(O(\text{fes})\) vertices (Theorem 4.5) |
|                        | lower bounds                         |                                 |
|                        | No kernel with size poly(vc + r) in \(K_{\ell,r}\)-free graphs (Theorem 2.14) | No kernel with size poly(tw + k + \ell) even in planar graphs with const. \(\Delta\) (Theorem 3.13) |
|                        | WK[1]-hard w. r. t. vc + r (Theorem 2.14) | No kernel with size poly(fvs + \ell) (Theorem 4.20) |

Figure 1.1: Overview on the existence of polynomial kernelization. Gray: no polynomial-size kernel unless the polynomial-time hierarchy collapses. White: polynomial-size kernel proved in this paper.

Almost planar and tree-like transportation networks. The focus of our work is two-fold. Firstly, since the problem is NP-hard, we search for efficient algorithms in graphs that are likely to occur as transportation networks: almost planar graphs, which occur as road networks, and tree-like graphs, which arise as waterways (ignoring the few man-made canals, natural river networks form forests [27]). Secondly, given the effect that preprocessing and data reduction had to fundamental routing problems like finding shortest paths [2], we study the possibilities of polynomial-time data reduction with provable performance guarantees for SSP.

In order to measure the running time of our algorithms with respect to the “degree of planarity” or the “tree-likeness” of a graph, as well as to analyze the power of data reduction algorithms, we employ parameterized complexity theory, which provides us with the concepts of fixed-parameter algorithms and problem kernelization [15, 19, 24, 44]. Fixed-parameter algorithms have recently been applied to numerous NP-hard routing problems [4, 5, 13, 18, 30–34, 46, 47]. In particular, they led to subexponential-time algorithms for fundamental NP-hard routing problems in planar graphs [37] and to algorithms that work efficiently on real-world data [5].

Fixed-parameter algorithms. The main idea of fixed-parameter algorithms is to accept the exponential running time seemingly inherent to solving NP-hard problems, yet to restrict the combinatorial explosion to a parameter of the problem, which can be small in applications. We call a problem fixed-parameter tractable if it can be solved in \(f(k) \cdot n^{O(1)}\) time on inputs of length \(n\) and some function \(f\) depending only on some parameter \(k\). In contrast to an algorithm that merely runs in polynomial time for fixed \(k\), say, in \(O(n^k)\) time, which is intractable even for small values of \(k\), fixed-parameter algorithms can solve NP-hard problems quickly if \(k\) is small.

Provably effective polynomial-time data reduction. Kernelization allows for provably effective polynomial-time data reduction. Note that a result of the form “our polynomial-time data reduction algorithm reduces the input size by at least one bit, preserving optimality of solutions” is impossible for NP-hard problems unless \(P = NP\). In contrast, a kernelization algorithm reduces a problem instance into an equivalent one (the problem kernel) whose size depends only (ideally polynomially) on some problem parameter. Problem kernelization has been successfully applied to obtain effective polynomial-time data reduction algorithms for many NP-hard problems [29, 38] and also led to techniques for proving the limits of polynomial-time data reduction [8, 9, 43].
1.1 Our contributions

We study SSP (and a weighted variant) in two main classes of graphs: almost planar graphs and tree-like graphs. We refer to Table 1.1 and Figure 1.1 for an overview of our main results.

Regarding almost planar graphs, we study graphs with small crossing number in Section 2. We show that (even the weighted version of) SSP is solvable in subexponential $2^{O(\sqrt{n})}$-time in graphs with constant crossing number. Moreover, we prove that SSP is not solvable in $2^{\omega(\sqrt{n})}$-time in planar graphs unless the Exponential Time Hypothesis fails. In $K_{r,r}$-free graphs, which comprise the graphs with crossing number $O(r^3)$ [45], we show a problem kernel for SSP with size $\nu C^{O(r)}$, where $\nu$ is the vertex cover number of the input graph. We prove that, unless the polynomial-time hierarchy collapses, there is no problem kernel of size polynomial in $\nu + r$. Moreover, we prove that, presumably [35], SSP does not even allow for Turing kernels with size polynomial in $\nu + r$; that is, we could not solve SSP in polynomial time even if we precomputed all answers to subproblems of size polynomial in $\nu + r$ and could look them up in constant time.

Regarding tree-like graphs, we first study graphs with small treewidth in Section 3: we prove that (even the weighted version of) SSP is solvable in $2^{O(n \cdot t)} \cdot t^2 \cdot n$ time in graphs of treewidth $tw$ and that there is no problem kernel with size polynomial in $tw$. Then, we study graphs with small feedback edge and vertex sets in Section 4. We show problem kernels with $O(fes)$ vertices or $O(fvs \cdot (k + \ell) \cdot \max\{fvs \cdot k, \ell\})$ vertices, where $fes$ is the feedback edge number and $fvs$ is the feedback vertex number of the input graph. We also prove that, unless the polynomial-time hierarchy collapses to the third level, the latter kernel cannot be improved to be of size polynomial in $fvs + \ell$ or $fvs + k$.

1.2 Related work

Luckow and Fluschnik [41] first defined SSP and analyzed its parameterized complexity with respect to the parameters $k$ and $\ell$. In contrast to their work, we study problem parameters that describe the structure of the input graphs, in particular those that are small in transportation networks.

Chechik et al. [14] introduced the similar secluded path problem, that, given an undirected graph $G = (V, E)$ with two designated vertices $s, t \in V$, vertex-weights $w : V \rightarrow \mathbb{N}$, and two integers $k, C \in \mathbb{N}$, asks whether there is an $s-t$-path $P$ such that the size of the closed neighborhood $|N_G[V(P)]| \leq k$ and the weight of the closed neighborhood $w(N_G[V(P)]) \leq C$. Fomin et al. [25] studied the parameterized complexity of the problem. In particular, they prove that secluded path admits problem kernels with size polynomial in $k$ and the feedback vertex number. On the negative side, they prove that secluded path does not admit problem kernels with size polynomial in the vertex cover number $\nu$. Our negative results on kernelization for SSP are significantly stronger: not only do we show that there is no problem kernel of size polynomial in $\nu + r$ even in bipartite $K_{r,r}$-free graphs, we also show that SSP is W[1]-hard parameterized by $\nu + r$.

Van Bevern et al. [6] studied several classical graph optimization problems in both the "secluded" (small closed neighborhood) and the "small secluded" (small set with small open neighborhood) variants. Amongst others, they prove that while finding a secluded $s-t$ separator with small closed neighborhood remains solvable in polynomial time, finding a small secluded $s-t$ separator is NP-complete.

Golovach et al. [28] studied the "small secluded" scenario for finding connected induced subgraphs with given properties. They prove that if the requested property is characterized through finitely many forbidden induced subgraphs, then the problem is fixed-parameter tractable when parameterized by the size $\ell$ of the open neighborhood. Their result obviously does not generalize to SSP, since SSP is NP-hard even for $\ell = 0$ [41].

1.3 Preliminaries

We use basic notation from graph theory [17] and parameterized algorithms [15, 19, 24, 44]. By $A \cup B$, we denote the union of two sets $A$ and $B$ when we emphasize that $A$ and $B$ are disjoint.

**Graph Theory.** We study simple, finite, undirected graphs $G = (V, E)$. We denote by $V(G) := V$ the set of vertices of $G$ and by $E(G) := E$ the set of edges of $G$. We denote $n := |V|$ and $m := |E|$. For any subset $U \subseteq V$ of vertices, we denote by $N_G(U) = \{w \in V \setminus U \mid \exists v \in U : \{v, w\} \in E\}$ the open neighborhood of $U$ in $G$. When the graph $G$ is clear from the context, we drop the subscript $G$. A set $U \subseteq V$ of vertices is a vertex cover if every edge in $E$ has an endpoint in $U$. The size of a minimum vertex cover is called vertex cover number $\nu(G)$ of $G$. A set $F \subseteq E$ of edges is a feedback edge set if the graph $(G, E \setminus F)$ is a forest. The minimum size of a feedback edge set in a connected
graph is \( m - n + 1 \) and is called the feedback edge number \( \text{fes}(G) \) of \( G \). A set \( V' \subseteq V \) of edges is a feedback vertex set if the graph \( G - V' := (V \setminus V', e \in E(G) \mid e \cap V' = \emptyset) \) is a forest. The minimum size of a feedback vertex set is called the feedback vertex number \( \text{fvs}(G) \) of \( G \). The crossing number \( \text{cr}(G) \) of \( G \) is the minimum number of crossings in any planar drawing of \( G \) (where only two edges are allowed to cross in each point). A path \( P = (V, E) \) is a graph with vertex set \( V = \{x_0, x_1, \ldots, x_p\} \) and edge set \( E = \{\{x_i, x_{i+1}\} \mid 0 \leq i < p\} \). We say that \( P \) is an \( x_0-x_p \)-path of length \( p \). We also refer to \( x_0, x_p \) as the end points of \( P \), and to all vertices \( V \setminus \{x_0, x_p\} \) as the inner vertices of \( P \).

### Parameterized Complexity Theory

Let \( \Sigma \) be a finite alphabet. A parameterized problem \( L \) is a subset \( L \subseteq \Sigma^* \times \mathbb{N} \). An instance \((x, k) \in \Sigma^* \times \mathbb{N}\) is a yes-instance for \( L \) if and only if \((x, k) \in L\). We call \( x \) the input and \( k \) the parameter.

**Definition 1.2** (fixed-parameter tractability, FPT). A parameterized problem \( L \subseteq \Sigma^* \times \mathbb{N} \) is fixed-parameter tractable if there is a fixed-parameter algorithm deciding \((x, k) \in L \) in time \( f(k) \cdot |x|^{O(1)} \). The complexity class \( \text{FPT} \) consists of all fixed-parameter tractable problems.

**Definition 1.3** (kernelization). Let \( L \subseteq \Sigma^* \times \mathbb{N} \) be a parameterized problem. A kernelization is an algorithm that maps any instance \((x, k) \in \Sigma^* \times \mathbb{N} \) to an instance \((x', k') \in \Sigma^* \times \mathbb{N} \) in poly(|x| + k) time such that

\[
\begin{align*}
(i) & \quad (x, k) \in L \iff (x', k') \in L', \text{ and} \\
(ii) & \quad |x'| + k' \leq f(k) \text{ for some computable function } f.
\end{align*}
\]

We call \((x', k') \) the problem kernel and \( f \) its size.

**Basic observations.** We may assume our input graph to be connected due to the following obviously correct and linear-time executable data reduction rule.

**Reduction Rule 1.4.** If \( G \) has more than one connected component, then delete all but the component containing both \( s \) and \( t \) or return no if such a component does not exist.

### 2 Graphs with small crossing number

Many transportation networks such as rail and street networks are planar or at least have a small crossing number—the minimum number of edge crossings in a plane drawing of a graph. Unfortunately, SSP remains NP-hard even in planar graphs with maximum degree four and \( \ell = 0 \) [41].

In this section, we present algorithms for SSP in graphs with constant crossing number. These in particular apply to planar graphs. First, in Section 2.1, we present a subexponential-time algorithm and a matching lower bound. Second, in Section 2.2, we present a provably effective data reduction algorithm. Finally, in Section 2.3, we show the limits of data reduction algorithms for SSP in graphs with small but non-constant crossing number.

#### 2.1 A subexponential-time algorithm

In this section, we describe how to solve SSP in subexponential time in graphs with constant crossing number. More precisely, we prove the following theorem.

**Theorem 2.1.** **Short Secluded Path** is solvable in \( 2^{O(\sqrt{n})} \) time on graphs with constant crossing number.

We will also prove a matching lower bound. To prove **Theorem 2.1**, we exploit that graphs with constant crossing number are \( H \)-minor free for some graph \( H \) and a general framework for subexponential-time algorithms for \( H \)-minor free graphs by Demaine and Hajiaghayi [16].

**Definition 2.2** (graph minor). A graph \( H \) is a minor of a graph \( G \) if \( H \) can be obtained from \( G \) by a sequence of vertex deletions, edge deletions, and edge contractions. If a graph \( G \) does not contain \( H \) as a minor, then \( G \) is said to be \( H \)-minor free.

Bokal et al. [12] showed that, if a graph \( G \) contains \( K_{r,r} \) as a minor, then the crossing number of \( G \) is \( \text{cr}(G) \geq \frac{1}{4}(r-2)^2 \).

Thus, any graph \( G \) is \( K_{r,r} \)-minor free for \( r > \sqrt{2\text{cr}(G)} + 2 \), which goes in line with the well-known fact that planar graphs are \( K_{3,3} \)-minor free [48]. Demaine and Hajiaghayi [16] showed that, for any graph \( H \), all \( H \)-minor free
graphs have treewidth $tw \in O(\sqrt{n})$. To prove Theorem 2.1, it thus remains to show that SSP is solvable in $2^{O(tw)} \cdot \text{poly}(n)$ time, which is the main technical work deferred to Section 3.1.

Complementing Theorem 2.1, we now show that, unless the Exponential Time Hypothesis (ETH) fails, Theorem 2.1 can be neither significantly improved in planar graphs nor generalized to general graphs.

**Hypothesis 2.3** (Exponential Time Hypothesis (ETH), Impagliazzo et al. [36]). There is a constant $c$ such that $n$-variable 3-SAT cannot be solved in $2^{c(n+m)}$ time.

The ETH was introduced by Impagliazzo et al. [36] and since then has been used to prove running time lower bounds for various NP-hard problems (we refer to Cygan et al. [15, Chapter 14] for an overview). We use it to prove the following theorem.

**Theorem 2.4.** Unless the Exponential Time Hypothesis fails, Short Secluded Path has no $2^{o(\sqrt{n})}$-time algorithm in planar graphs and no $2^{o(n+m)}$-time algorithm in general.

**Proof.** Assume that there is a $2^{o(\sqrt{n})}$-time algorithm for SSP in planar graphs and a $2^{o(n)}$-time algorithm for SSP in general graphs. Luckow and Fluschnik [41] give a polynomial-time many-one reduction from Hamiltonian Cycle to SSP that maintains planarity and increases the number of vertices and edges by at most a constant. Thus, we get a $2^{o(\sqrt{n})}$-time algorithm for Hamiltonian Cycle in planar graphs and a $2^{o(n+m)}$-time algorithm in general graphs. This contradicts ETH [15, Theorems 14.6 and 14.9]. □

### 2.2 Effective data reduction

In the previous section, we have shown a subexponential-time algorithm for SSP in graphs with constant crossing number. There, we exploited the fact that graphs with crossing number $cr$ are $K_{cr}$-minor free for $r > \sqrt{2cr} + 2$. Of course, this means that they neither contain $K_{cr}$ as subgraph (indeed, one can show this even for $r \geq 3.145 \cdot \sqrt{cr}$ using bounds from Pach et al. [45]).

In this section, we show a data reduction algorithm that reduces any instance of SSP in $K_{cr}$-free graphs to an equivalent instance with size polynomial in the vertex cover number of the input graph. In the next section, we prove that it does not generalize to general graphs.

**Theorem 2.5.** For each constant $r \in \mathbb{N}$, Short Secluded Path in $K_{cr}$-free graphs admits a problem kernel with size polynomial in the vertex cover number of the input graph.

The proof of Theorem 2.5 consists of three steps. First, in linear time, we transform an $n$-vertex instance of SSP into an equivalent instance of an auxiliary vertex-weighted version of SSP with $O(vc')$ vertices. Second, using a theorem of Frank and Tardos [26], in polynomial time, we reduce the vertex weights to $2^{O(vc')}$ so that the length of their encoding is $O(vc'^2)$. Finally, since SSP is NP-complete in planar, and, hence, in $K_{3,3}$-free graphs, we can, in polynomial time, reduce the shrunk instance back to an instance of the unweighted SSP in $K_{cr}$-free graphs. Due to the polynomial running time of the reduction, there is at most a polynomial blow-up of the instance size.

Our auxiliary variant of SSP allows each vertex to have three weights: weight $κ(v)$ counts towards the length of the path, the weights $λ(v)$ and $η(v)$ count towards the number of neighbors (in fact, we will not use $η(v)$ yet, but to derive other results later).

**Problem 2.6** (Vertex-Weighted Short Secluded Path (VW-SSP)).

**Instance:** An undirected, simple graph $G = (V,E)$ with two distinct vertices $s, t \in V$, two integers $k \geq 2$ and $ℓ \geq 0$, and vertex weights $κ : V \to \mathbb{N}$, $λ : V \to \mathbb{N} \cup \{0\}$, and $η : V \to \mathbb{N} \cup \{0\}$.

**Question:** Is there a simple $s$-$t$-path $P$ with $\sum_{v \in (P)} κ(v) \leq k$ and $\sum_{v \in (P)} λ(v) + \sum_{v \in N(V(P))} η(v) + \sum_{v \in N(V(P))} λ(v) \leq ℓ$ in $G$?

Note that an instance of SSP can be considered to be an instance of VW-SSP with unit weight functions $κ$ and $λ$ and the zero weight function $η$. Our data reduction will be based on removing twins.

**Definition 2.7** (twins). Two vertices $u$ and $v$ are called (false) twins if $N(u) = N(v)$.

As the first step towards proving Theorem 2.5, we will show that the following data reduction rule, when applied to an $K_{cr}$-free instance of SSP for constant $r$, leaves us with an instance of VW-SSP with $O(vc')$ vertices.

---

1 In fact, they showed $tw \in O(\sqrt{n})$ for any graph parameter $q$ that is $Ω(p)$ on a $(\sqrt{p} \times \sqrt{p})$-grid and does not increase when taking minors. For example, the vertex cover number or feedback vertex number.
Reduction Rule 2.8. Let \((G, s, t, k, \ell, \kappa, \lambda, \eta)\) be an \(\text{SSP}\) instance with unit weights \(\kappa\) and \(\lambda\), and zero weights \(\eta\), where \(G = (V, E)\) is a \(K_{r, r}\)-free graph.

For each maximal set \(U \subseteq V \setminus \{s, t\}\) of twins such that \(|U| > r\), delete \(|U| - r + 1\) vertices of \(U\) from \(G\), and, for an arbitrary remaining vertex \(v \in U\), set \(\lambda(v) := |U| - r\) and \(\kappa(v) := k + 1\).

Lemma 2.9. Reduction Rule 2.8 is correct and can be applied in linear time.

Proof. We first show the linear running time. A module \(U\) of a graph is a maximal set of vertices such that \(N(u) \setminus U = N(v) \setminus U\) for each \(u, v \in U\). The modules of a graph are computable in linear time [42]. From these modules, we obtain a partition of \(V\) into maximal sets of twins by taking the modules that are themselves independent sets. It is now easy to check which of them has size larger than \(r\) and to apply Reduction Rule 2.8.

To prove that Reduction Rule 2.8 is correct, we prove that its input instance \(I = (G, s, t, k, \ell, \kappa, \lambda, \eta)\) is a yes-instance if and only if its output instance \(I' = (G', s, t, k, \ell', \kappa', \lambda', \eta)\) is. Herein, note that \(\eta\) is the zero function, so we will ignore it in the rest of the proof.

\(\Rightarrow\) Let \(P\) be a simple \(s\)-\(t\)-path with \(\sum_{v \in V(P)} \kappa(v) \leq k\) and \(\sum_{v \in N(V(P))} \lambda(v) \leq \ell\) in \(G\). Let \(U \subseteq V \setminus \{s, t\}\) be an arbitrary set of twins with \(|U| > r\). Since \(G\) is \(K_{r, r}\)-free, \(|N(U)| \leq r - 1\). Thus, \(P\) contains at most \(|N(U)| - 1 = r - 1\) vertices of \(U\). Reduction Rule 2.8 reduces \(U\) to a set \(U'\) with \(r\) vertices, where only one of the vertices \(v \in U'\) has weight \(\kappa'(v) > 1\). Thus, without loss of generality, we can assume that \(P\) uses only the \(r - 1\) vertices \(v \in U \cap U'\) with \(\kappa'(v) = 1\). Hence,

\[
\sum_{v \in V(P) \cap U} \kappa(v) = \sum_{v \in V(P) \cap U} \kappa'(v) = |V(P) \cap U|.
\]

(2.1)

Moreover, if \(P\) uses a vertex of \(U\), then it also uses a vertex of \(N(U)\) and, hence, \(U \setminus V(P) \subseteq N(V(P))\). Thus,

\[
\sum_{v \in N_G(V(P)) \cap U} \lambda(v) = \sum_{v \in U \setminus V(P)} \lambda(v) = \sum_{v \in V(P) \setminus U} \lambda'(v)
\]

(2.2)

since \(|U \setminus U'| = |U| - r\) and there is a vertex \(v \in U' \setminus U\) that has \(\lambda(v) = 1\) on the left-hand side of (2.2) but \(\lambda'(v) = |U| - r\) on the right-hand side of (2.2). From (2.1), (2.2), and the arbitrary choice of \(U\), it follows that \(P\) is an \(s\)-\(t\)-path with \(\sum_{v \in V(P)} \kappa'(v) \leq k\) and \(\sum_{v \in N(V(P))} \lambda'(v) \leq \ell\) in \(G'\). Thus, \(I'\) is a yes-instance.

\(\Leftarrow\) Let \(P\) be a simple \(s\)-\(t\)-path with \(\sum_{v \in V(P)} \kappa(v) \leq k\) and \(\sum_{v \in N(V(P))} \lambda(v) \leq \ell\) in \(G'\). Let \(U \subseteq V \setminus \{s, t\}\) be a set of twins in \(G\) reduced to a subset \(U'\) in \(G'\) by Reduction Rule 2.8. The only vertex \(v \in U'\) with weight \(\kappa'(v) > 1 = \kappa(v)\) has \(\kappa'(v) = k + 1\) and thus is not on \(P\). Yet, if \(P\) uses vertices of \(U'\), then \(v \in U' \setminus V(P) \subseteq N_G(V(P))\) and \(U \setminus V(P) \subseteq N_G(V(P))\). Thus, (2.1) and (2.2) apply and, together with the arbitrary choice of \(U\), show that \(P\) is an \(s\)-\(t\)-path with \(\sum_{v \in V(P)} \kappa(v) \leq k\) and \(\sum_{v \in N(V(P))} \lambda(v) \leq \ell\) in \(G\) and, thus, \(I\) is a yes-instance.

Having proved the correctness of Reduction Rule 2.8, we now prove a size bound for the instances remaining after Theorem 2.5.

Proposition 2.10. Applied to an instance of \(\text{SSP}\) with an \(K_{r, r}\)-free graph with vertex cover number \(vc\), Reduction Rules 2.1 and 2.8 yield an instance of \(\text{SSP}\) on at most \((vc + 2) + r(vc + 2)^r\) vertices in linear time.

Proof. Let \((G', s, t, k, \ell', \kappa', \lambda', \eta)\) be the instance obtained from applying Reduction Rules 2.1 and 2.8 to an instance \((G, s, t, k, \ell, \kappa, \lambda, \eta)\).

Let \(C\) be a minimum-cardinality vertex cover for \(G'\) that contains \(s\) and \(t\), and let the vertex set of \(G'\) be \(V = C \cup Y\).

Since \(G'\) is a subgraph of \(G\), one has \(|C| \leq vc(G') + 2 \leq vc(G) + 2 = vc + 2\). It remains to bound \(|Y|\). To this end, we bound the number of vertices of degree at least \(r\) in \(Y\) and the number of vertices of degree exactly \(i\) in \(Y\) for each \(i \in \{0, \ldots, r - 1\}\). Note that vertices in \(Y\) have neighbors only in \(C\).

Since Reduction Rule 1.4 has been applied, there are no vertices of degree zero in \(Y\).

Since Reduction Rule 2.8 has been applied, for each \(i \in \{1, \ldots, r - 1\}\) and each subset \(C' \subseteq C\) with \(|C'| = i\), we find at most \(r\) vertices in \(Y\) whose neighborhood is \(C'\). Thus, for each \(i \in \{1, \ldots, r - 1\}\), the number of vertices with degree \(i\) in \(Y\) is at most \(r \cdot \binom{|C|}{i}\).

Finally, since \(G\) is \(K_{r, r}\)-free, any \(r\)-sized subset of the vertex cover \(C\) has at most \(r - 1\) common neighbors. Hence, since vertices in \(Y\) have neighbors only in \(C\), the number of vertices in \(Y\) of degree greater or equal to \(r\) is at most \((r - 1) \cdot \binom{|C|}{r}\).

We conclude that

\[
|Y| \leq |C| + (r - 1) \cdot \binom{|C|}{r} + r \cdot \sum_{i=1}^{r-1} \binom{|C|}{i} \leq (vc + 2) + r(vc + 2)^r.
\]

\[\square\]
Having shown how to reduce an instance of SSP on $K_{r,r}$-free graphs to an equivalent instance of VW-SSP on $O(vc^3)$ vertices for constant $r$, we finished the first step to proving Theorem 2.5. However, our data reduction works by “hiding” an unbounded number of twins in vertices of unbounded weights. Therefore, the second step on the proof of Theorem 2.5 is reducing the weights.

To reduce the weights of an VW-SSP instance, we are going to apply a theorem by Frank and Tardos [26]. The theorem is a key approach to polynomial-size kernels for weighted problems [20]. Notably, we are seemingly the first ones to apply the theorem of Frank and Tardos [26] to eventually kernelize an unweighted problem—SSP.

Proposition 2.11 (Frank and Tardos [26]). There is an algorithm that, on input $w \in \mathbb{Q}^d$ and integer $N$, computes in polynomial time to an instance $I$ of VW-SSP with $\|w\|_\infty \leq 2^d N^{d(d+2)}$ such that $\text{sign}(w^T b) = \text{sign}(\bar{w}^T b)$ for all $b \in \mathbb{Z}^d$ with $\|b\|_1 \leq N - 1$, where

$$
\text{sign}(x) = \begin{cases} 
+1 & \text{if } x > 0, \\
0 & \text{if } x = 0, \text{ and} \\
-1 & \text{if } x < 0.
\end{cases}
$$

Observation 2.12. For $N \geq 2$, Proposition 2.11 gives $\text{sign}(w^T e_i) = \text{sign}(\bar{w}^T e_i)$ for each $i \in \{1, \ldots, d\}$, where $e_i \in \mathbb{Z}^d$ is the vector that has 1 in the $i$-th coordinate and zeroes in the others. Thus, one has $\text{sign}(w_i) = \text{sign}(\bar{w}_i)$ for each $i \in \{1, \ldots, d\}$. That is, when reducing a weight vector from $w$ to $\bar{w}$, Proposition 2.11 maintains the signs of weights.

We apply Proposition 2.11 and Observation 2.12 to the weights of VW-SSP.

Lemma 2.13. An instance $I = (G, s, t, k, \ell, r, \kappa, \eta)$ of VW-SSP on an $n$-vertex graph $G = (V, E)$ can be reduced in polynomial time to an instance $I' = (G, s, t, k', \ell', r', \kappa', \eta')$ of VW-SSP such that

i) $|V', \kappa'(v), \ell'(v), \eta'(v)| \subseteq \{0, \ldots, 2^{(n + 1)^3}, (n + 2)^{(2n + 1)(2n + 3)}\}$, for each vertex $v \in V$, and

ii) $I$ is a yes-instance if and only if $I'$ is a yes-instance.

Proof. In this proof, we will conveniently denote the weight functions $\lambda, \lambda', \kappa, \kappa', \eta, \text{and } \eta'$ as vectors in $\mathbb{H}^n$ such that $\lambda_v = \lambda(v)$ for each $v \in V$, and similarly for the other weight functions.

We apply Proposition 2.11 with $d = 2n + 1$ and $N = n + 2$ separately to the vectors $(\eta, \lambda, \ell) \in \mathbb{H}^{2n + 1}$ and $(\kappa, \{0\}^n, k) \in \mathbb{H}^{2n + 1}$ to obtain vectors $(\eta', \lambda', \ell') \in \mathbb{Z}^{2n + 1}$ and $(\kappa', \{0\}^n, k') \in \mathbb{Z}^{2n + 1}$ in polynomial time.

(i) This follows from Proposition 2.11 with $d = 2n + 1$ and $N = n + 2$, and from Observation 2.12 since $(\eta, \lambda, \ell)$ and $(\kappa, \{0\}^n, k)$ are vectors of nonnegative numbers.

(ii) Consider an arbitrary $s$-$t$-path $P$ in $G$ and two associated vectors $x, y \in \mathbb{Z}^n$, where

$$
x_v = \begin{cases} 
1 & \text{if } v \in V(P), \\
0 & \text{otherwise},
\end{cases} \quad y_v = \begin{cases} 
1 & \text{if } v \in N(V(P)) \text{ and} \\
0 & \text{otherwise}.
\end{cases}
$$

Observe that $\|(x, y, -1)\|_1 \leq n + 1$ and $\|(x, \{0\}^n, -1)\|_1 \leq n + 1$. Since $n + 1 \leq N - 1$, Proposition 2.11 gives

$$
\text{sign}((x, y, -1)^T(\eta, \lambda, \ell)) = \text{sign}((x, y, -1)^T(\eta', \lambda', \ell')) \quad \text{and} \quad \text{sign}((x, \{0\}^n, -1)^T(\kappa, \{0\}^n, k)) = \text{sign}((x, \{0\}^n, -1)^T(\kappa', \{0\}^n, k')),
$$

which is equivalent to

$$
\sum_{v \in V(P)} \eta(v) + \sum_{v \in N(V(P))} \lambda(v) \leq \ell \quad \iff \quad \sum_{v \in V(P)} \eta'(v) + \sum_{v \in N(V(P))} \lambda'(v) \leq \ell' \quad \text{and} \quad \sum_{v \in P} \kappa(v) \leq k \iff \sum_{v \in P} \kappa'(v) \leq k'.
$$

We have finished two steps towards the proof of Theorem 2.5: we reduced SSP in $K_{r,r}$-free graphs for constant $r$ to instances of VW-SSP with $O(vc^3)$ vertices using Proposition 2.10 and shrunk its weights to encoding-length $O(vc^3)$ using Lemma 2.13. To finish the proof of Theorem 2.5, it remains to reduce VW-SSP back to SSP on $K_{r,r}$-free graphs.
Proof of Theorem 2.5. Using Proposition 2.10 and Lemma 2.13, we reduce any SSP instance $I$ on a $K_{r,r}$-free $n$-vertex graph for constant $r$ with vertex cover number $vc$ to an equivalent VW-SSP instance $I'$ on $O(vc^\ell)$ vertices whose weights are bounded by $2^{O(vc^\ell)}$. Thus, the overall encoding length of $I'$ is $O(vc^\ell)$. Since SSP is NP-complete even in planar graphs [41] and, therefore, in $K_{3,3}$-free graphs, we can in polynomial time reduce $I'$ to an equivalent instance $I''$ of SSP on $K_{r,r'}$-free graphs. Since the running time of the reduction is polynomial, the size of $I''$ is polynomial in the size of $I'$ and, hence, polynomial in $vc$. \hfill $\Box$

2.3 Limits of data reduction

In Section 2.2, we have seen that SSP allows for problem kernels with size polynomial in $vc$ if the input graph is $K_{r,r}$-free for some constant $r$. A natural question is whether one can loosen the requirement of $r$ being constant.

The following Theorem 2.14(i) shows that, under reasonable complexity-theoretic assumptions, this is not the case: we cannot get problem kernels whose size bound depends polynomially on both $vc$ and $r$. Moreover, the following Theorem 2.14(ii) shows that, presumably [35], SSP does not even have Turing kernels with size polynomial in $vc + r$. That is, we could not even solve SSP in polynomial time if we had precomputed all answers to SSP instances with size polynomial in $vc + r$ and could look them up in constant time.

Both results come surprisingly: finding a standard shortest $s$-$t$-path is easy, whereas finding a short secluded path in general graphs is so hard that not even preprocessing helps.

Theorem 2.14. Even in bipartite graphs, Short Secluded Path

\begin{enumerate}
\item[i)] has no problem kernel with size polynomial in $vc + r$ unless $coNP \subseteq NP/poly$ and
\item[ii)] is WK[1]-hard when parameterized by $vc + r$,
\end{enumerate}

where $vc$ is the vertex cover number of the input graph and $r$ is the smallest number such that the input graph is $K_{r,r}$-free.

In the following, we prove Theorem 2.14. The proof exploits that Multicolored Clique is WK[1]-hard parameterized by $k \log n$ [35]:

Problem 2.15 (Multicolored Clique).

Instance: A $k$-partite $n$-vertex graph $G = (V,E)$, where $V = \bigcup_{i=1}^{k} V_i$ for independent sets $V_i$.

Question: Does $G$ contain a clique of size $k$?

We use the following type of reduction, which transfers WK[1]-hardness from Multicolored Clique to SSP [35]:

Definition 2.16 (polynomial parameter transformation). Let $L, L' \subseteq \Sigma^* \times \mathbb{N}$ be two parameterized problems. A polynomial parameter transformation from $L$ to $L'$ is an algorithm that maps any instance $(x,k) \in \Sigma^* \times \mathbb{N}$ to an instance $(x', k') \in \Sigma^* \times \mathbb{N}$ in $\mathit{poly}(|x| + k)$ time such that

\begin{enumerate}
\item [(i)] $(x,k) \in L \iff (x',k') \in L'$, and
\item [(ii)] $k' \leq \mathit{poly}(k)$.
\end{enumerate}

Our polynomial parameter transformation of Multicolored Clique into SSP uses the following gadget.

Definition 2.17 ($z$-binary gadget). A $z$-binary gadget for some power $z$ of two is a set $B = \{u_1, u_2, \ldots, u_{2\log z}\}$ of vertices. We say that a vertex $v$ is $p$-connected to $B$ for some $p \in \{0, \ldots, z-1\}$ if $v$ is adjacent to $u_q \in B$ if and only if there is a “1” in position $q$ of the string that consists of the binary encoding of $p$ followed by its complement.

Example 2.18. The binary encoding of $5$ followed by its complement is $101010$. Thus, a vertex $v$ is $5$-connected to an $8$-binary gadget $\{u_1, \ldots, u_8\}$ if and only if $v$ is adjacent to $u_1, u_3,$ and $u_5$. Also observe that, if a vertex $v$ is $q$-connected to a $z$-binary gadget $B$, then $v$ is adjacent to exactly half of the vertices of $B$, that is, to $\log z$ vertices of $B$.

The following reduction from Multicolored Clique to SSP is illustrated in Figure 2.1.
Construction 2.19. Let \( G = (V, E) \) be an instance of Multicolored Clique, where \(|V| = n\) and \( V = V_1 \uplus V_2 \uplus \cdots \uplus V_k \). Without loss of generality, assume that \( V_i = \{v^i_1, v^i_2, \ldots, v^i_n\} \) for each \( i \in \{1, \ldots, k\} \), where \( n \) is some power of two (we can guarantee this by adding isolated vertices to \( G \)). We construct an equivalent instance \((G', s, t, k', \ell')\) of SSP, where

\[
k' := \left(\frac{k}{2}\right) + 1, \quad \ell' := |E| - \left(\frac{k}{2}\right) + k \log n,
\]

and the graph \( G' = (V', E') \) is as follows. The vertex set \( V' \) consists of vertices \( s, t, \) a vertex \( v_e \) for each edge \( e \in E \), vertices \( w_h \) for \( h \in \{1, \ldots, \left(\frac{k}{2}\right) - 1\} \), and mutually disjoint \( n \)-binary vertex gadgets \( B_1, \ldots, B_k \), each vertex in which has \( \ell' + 1 \) neighbors of degree one. We denote

\[
E' := \{v_e \in V' \mid e \in E\}, \quad B := B_1 \uplus B_2 \uplus \cdots \uplus B_k, \quad \text{and} \quad W := \{w_h \mid 1 \leq h \leq \left(\frac{k}{2}\right) - 1\}.
\]

The edges of \( G' \) are as follows. For each edge \( e = \{v^i_j, v^j_i\} \in E \), vertex \( v_e \in E_{ij} \) of \( G' \) is \( p \)-connected to \( B_i \) and \( q \)-connected to \( B_j \). Vertex \( s \in V' \) is adjacent to all vertices in \( E_{1,2} \) and vertex \( t \in V' \) is adjacent to all vertices in \( E_{k-1,k} \). Finally, to describe the edges incident to vertices in \( W \), consider any ordering of pairs \( \{(i, j) \mid 1 \leq i < j \leq k\} \). Then, vertex \( w_h \in W \) is adjacent to all vertices in \( E_{ij} \) and to all vertices in \( E_{i'j'} \), where \( (i, j) \) is the \( h \)-th pair in the ordering and \( (i', j') \) is the \((h + 1)\)-st. This finishes the construction.

On our way proving Theorem 2.14, we aim to prove that Construction 2.19 is a polynomial-time many-one reduction that generates bipartite \( K_r \)-free graphs for sufficiently small \( r \) with sufficiently small vertex covers.

Lemma 2.20. The graph created by Construction 2.19 from an \( n \)-vertex instance \( G = (V_1 \uplus V_2 \uplus \cdots \uplus V_k, E) \) of Multicolored Clique is bipartite, \( K_{r,r} \)-free for \( r := 2k \log n + \left(\frac{k}{2}\right) + 2 \), and admits a vertex cover of size \( r - 1 \).

Proof. The constructed graph \( G' = (V', E') \) is bipartite with \( V' = X \uplus Y \), where

\[
X = \{s, t\} \cup W \cup B \quad \text{and} \quad Y = N(B) \cup E'.
\]

Hence, \( X \) is a vertex cover of size at most \( r - 1 \) in \( G' \). Finally, consider any \( K_{r,r} \) whose vertex set is partitioned into two independent sets \( X' \uplus Y' \subseteq V' \). Since \(|X'| = |Y'| = r\), \( |X' \cap X| \leq r - 1 \), and \( |Y' \cap X| \leq r - 1 \), we find \( u \in X' \cap Y \) and \( v \in Y' \cap Y \). Observe that \( \{u, v\} \) is an edge in the \( K_{r,r} \) but not in \( G' \). Thus, the \( K_{r,r} \) is not a subgraph of \( G' \). \( \square \)

Lemma 2.21. Construction 2.19 is a polynomial parameter transformation of Multicolored Clique parameterized by \( k \log n \) into SSP in \( K_{r,r} \)-free graphs parameterized by \( \text{vc} + r \).
Proof. Let \( I' := (G', s, t, k', \ell') \) be the SSP instance created by Construction 2.19 from an Multicolored Clique instance \( G = (V, E) \). In Lemma 2.20, we already showed \( vc + r \in poly(k \log n) \). Thus, it remains to show that \( G \) is a yes-instance if and only if \( I' \) is.

\((\Rightarrow)\) Let \( C \) be the edge set of a clique of size \( k \) in \( G \). For each \( 1 \leq i < j \leq k \), \( C \) contains exactly one edge \( e \) between \( V_i \) and \( V_j \). Thus, \( E_C := \{e \in E^+ \mid e \in C\} \) is a set of \( \binom{k}{2} \) vertices—exactly one vertex of \( E_{ij} \) for each \( 1 \leq i < j \leq k \). Thus, by Construction 2.19, \( G' \) contains an \( s,t \)-path \( P = (V_p, E_p) \) with \( |V_p| \leq k' \): its inner vertices are \( E_C \cup W \), alternating between these two sets. To show that \( (G', s, t, k', \ell') \) is a yes-instance, it remains to show \( |N(V_p)| \leq \ell' \).

Since \( P \) contains all vertices of \( W \), one has \( N(V_p) \subseteq B \cup (E^+ \setminus E_C) \), where \( |E^+ \setminus E_C| = |E| - \binom{k}{2} \). To show \( |N(V_p)| \leq \ell' \), it remains to show that \( |N(V_p) \cap B| \leq k \log n \). To this end, we show that \( |N(V_p) \cap B| = \log n \) for each \( i \in [1, \ldots, k] \).

The vertices in \( W \cup [s, t] \) have no neighbors in \( B \). Thus, consider arbitrary vertices \( v_{v_1}, v_{v_2} \in E_C \) such that \( N(v_{v_1}) \cap B_i \neq \emptyset \) and \( N(v_{v_2}) \cap B_i \neq \emptyset \) for some \( i \in [1, \ldots, k] \) (possibly, \( e_1 = e_2 \)). Then, \( e_1 = \{v_{v_1}, v_{v_2}\} \) and \( e_2 = \{v_{v_2}^i, v_{v_2}^j\} \). Since \( C \) is a clique, \( e_1 \) and \( e_2 \) are incident to the same vertex of \( V_i \). Thus, we have \( p = p' \). Both \( v_1 \) and \( v_2 \) are therefore \( p \)-connected to \( B_i \) and hence have the same \( \log n \) neighbors in \( B_i \). It follows that \( N(V_p) \leq \ell' \) and, consequently, that \( P \) is a yes-instance.

\((\Leftarrow)\) Let \( P = (V_p, E_p) \) be an \( s,t \)-path in \( G' \) with \( |V_p| \leq k' \) and \( |N(V_p)| \leq \ell' \). The path \( P \) does not contain any vertex of \( B \), since each of them has \( \ell' + 1 \) neighbors of degree one. Thus, the inner vertices of \( P \) alternate between vertices in \( W \) and in \( E^+ \). We get \( N(V_p) = (E^+ \setminus V_p) \cup (N(V_p) \cap B) \). Since \( P \) contains one vertex of \( E_{ij} \) for each \( 1 \leq i < j \leq k \), we know \( |N(V_p)| \leq |E| - \binom{k}{2} \). Thus, since \( |N(V_p)| \leq \ell' \), we have \( |N(V_p) \cap B| \leq k \log n \). We exploit this to show that the set \( C := \{e \in E \mid v \in V_p \cap E^+\} \) is the edge set of a clique in \( G \). To this end, it is enough to show that, for each \( i \in [1, \ldots, k] \), any two edges \( e_1, e_2 \in C \) with \( e_1 \cap V_i \neq \emptyset \) and \( e_2 \cap V_i \neq \emptyset \) have the same endpoint in \( V_i \): then \( C \) is a set of \( \binom{k}{2} \) edges on \( k \) vertices and thus forms a \( k \)-clique.

For each \( 1 \leq i < j \leq k \), \( P \) contains exactly one vertex \( v \in E_{ij} \), which has exactly \( \log n \) neighbors in each of \( B_i \) and \( B_j \). Thus, from \( |N(V_p) \cap B| \leq k \log n \) follows \( |N(V_p) \cap B| = \log n \) for each \( i \in [1, \ldots, k] \). It follows that, if two vertices \( v_{v_1} \) and \( v_{v_2} \) on \( P \) both have neighbors in \( B_i \), then both are \( \ell \)-connected to \( B_i \) for some \( p \), which means that the edges \( e_1 \) and \( e_2 \) of \( G \) share endpoint \( v_{v_2}^j \).

We conclude that \( C \) is the edge set of a clique of size \( k \) in \( G \). Hence, \( G \) is a yes-instance. \( \square \)

To prove Theorem 2.14, it is know a matter of putting together Lemma 2.21 and the fact that Multicolored Clique parameterized by \( k \log n \) is \( \text{W}[1] \)-complete.

Proof of Theorem 2.14. By Lemma 2.21, Construction 2.19 is a polynomial parameter transformation from Multicolored Clique parameterized by \( k \log n \) to SSP parameterized by \( vc + r \) in \( K_{q,r} \)-free graphs. Multicolored Clique parameterized by \( k \log n \) is known to be \( \text{W}[1] \)-complete [35] and hence, does not admit a polynomial-size problem kernel unless \( \text{coNP} \subseteq \text{NP}/\text{pol} \). From the polynomial parameter transformation in Construction 2.19, it thus follows that SSP is \( \text{W}[1] \)-hard parameterized by \( vc + r \) and does not admit a polynomial-size problem kernel unless \( \text{coNP} \subseteq \text{NP}/\text{pol} \).

\section{3 Graphs with small treewidth}

In this section, we provide algorithms for SSP in tree-like graphs. Such graphs naturally arise as waterways: when ignoring the few man-made canals, the remaining, natural waterways usually form a forest [27].

Moreover, graphs of small treewidth (formally defined in Section 3.1) are interesting since, as described in Section 2.1, graphs with constant crossing number have treewidth at most \( \sqrt{q} \) for many graph parameters \( q \). Thus, one can derive subexponential-time algorithms for these parameters from single-exponential algorithms for treewidth, like we did in Section 2.1.

First, in Section 3.1, we present an algorithm that efficiently solves SSP on graphs of small treewidth. Then, in Section 3.2, we prove that SSP allows for no problem kernel with size polynomial in the treewidth of the input graph.

### 3.1 A fixed-parameter algorithm

In this section, we provide a \( 2^{O(tw)} \cdot \ell^2 \cdot n \)-time algorithm for SSP in graphs of treewidth \( tw \), which will also conclude the proof of the \( 2^{O(\sqrt{\log(n)})} \)-time algorithm for SSP in graphs with constant crossing number (Theorem 2.1).
Before describing the algorithm, we formally introduce the treewidth concept. We will roughly follow the notation for tree decompositions of Bodlaender et al. [10], since we will be using some of their results to make our algorithm run in single-exponential time.

**Definition 3.1** (tree decomposition, treewidth). A tree decomposition \( \mathcal{T} = (T, \beta) \) of a graph \( G = (V, E) \) consists of a tree \( T \) and a function \( \beta : V(T) \to 2^V \) that associates each node \( x \) of the tree \( T \) with a subset \( B_x := \beta(x) \subseteq V \), called a bag, such that

i) for each vertex \( v \in V \), there is a node \( x \) of \( T \) with \( v \in B_x 

ii) for each edge \( \{u, v\} \in E \), there is a node \( x \) of \( T \) with \( \{u, v\} \subseteq B_x 

iii) for each node \( v \) of \( T \) with \( \beta(v) \neq \emptyset \), there is a node \( x \) of \( T \) with \( \beta(v) = B_x \). This node is called a node labeled with an edge \( \{u, v\} \).

The width of \( \mathcal{T} \) is \( w(\mathcal{T}) := \max_{x \in V(T)}|B_x| - 1 \). The treewidth of \( G \) is \( tw(G) := \min\{w(\mathcal{T}) | \mathcal{T} \text{ is a tree decomposition of } G \} \).


In this section, we will prove the following result.

**Theorem 3.2.** Short secluded path is solvable in \( 2^{2\omega\cdot \ell^2} \cdot n^2 \cdot n \) time in graphs of treewidth \( tw \).

To prove Theorem 3.2, we first need to compute a tree decomposition of the input graph. Using the classical algorithm of Bodlaender [7], a tree decomposition of minimum width \( tw(G) \) for a graph \( G \) can be computed in \( f(tw(G)) \cdot n \) time. However, the running time of this algorithm is not singly exponential in \( tw \), and thus too large for Theorem 3.2. Bodlaender et al. [11] proved that a tree decomposition of width \( O(tw(G)) \) of a graph \( G \) can be computed in \( 2^{2\omega \ell^2} \cdot n \)-time. Applying the following Proposition 3.3 to such a tree decomposition yields Theorem 3.2:

**Proposition 3.3.** Vertex-weighted short secluded path is solvable in \( n \cdot \ell^2 \cdot tw^{O(1)} \cdot (2 + 12 \cdot 2^\omega t)^n \) time when a tree decomposition of width \( tw \) is given, where \( \omega < 2.2373 \) is the matrix multiplication exponent.

To prove Theorem 3.2, it thus remains to prove Proposition 3.3. Note that Proposition 3.3 actually solves the weighted problem VW-SSP (Problem 2.6), where the term \( \ell^2 \) is only pseudo-polynomial for VW-SSP. It is a true polynomial for SSP since we can assume \( \ell \leq n \).

3.1.1 Assumptions on the tree decomposition

Our algorithm for Proposition 3.3 will work on the following simplified kind of tree decomposition, which can be obtained from a classical tree decomposition of width \( tw \) in \( n \cdot tw^{O(1)} \) time without increasing its width [10].

**Definition 3.4** (nice tree decomposition). A nice tree decomposition \( \mathcal{T} \) is a tree decomposition with one special bag \( r \) called the root and in which each bag is of one of the following types.

**Leaf node:** a leaf \( x \) of \( \mathcal{T} \) with \( B_x = \emptyset \).

**Introduce vertex node:** an internal node \( x \) of \( \mathcal{T} \) with one child \( y \) such that \( B_x = B_y \cup \{v\} \) for some vertex \( v \notin B_y \). This node is said to introduce vertex \( v \).

**Introduce edge node:** an internal node \( x \) of \( \mathcal{T} \) labeled with an edge \( \{u, v\} \in E \) and with one child \( y \) such that \( \{u, v\} \subseteq B_x = B_y \). This node is said to introduce edge \( \{u, v\} \).

**Forget node:** an internal node \( x \) of \( \mathcal{T} \) with one child \( y \) such that \( B_x = B_y \setminus \{v\} \) for some node \( v \in B_y \). This node is said to forget \( v \).

**Join node:** an internal node \( x \) of \( \mathcal{T} \) with two children \( y \) and \( z \) such that \( B_x = B_y = B_z \).

We additionally require that each edge is introduced at most once and make the following, problem specific assumptions on tree decompositions.

**Assumption 3.5.** When solving VW-SSP, we will assume that the source \( s \) and destination \( t \) of the sought path are contained in all bags of the tree decomposition and that the root bag contains only \( s \) and \( t \). This ensures that

- every bag contains vertices of the sought solution, and that
Figure 3.1: Illustration of a partial solution: the thick edges are an overall solution, where the darker edges are the part of the solution in $G_x$. The dashed edges are forbidden to exist.

- $s$ and $t$ are never forgotten or introduced.

Such a tree decomposition can be obtained from a nice tree decomposition by rooting it at a leaf (an empty bag) and adding $s$ and $t$ to all bags. This will increase the width of the tree decomposition by at most two.

Our algorithm will be based on computing partial solutions for subgraphs induced by a node of a tree decomposition by means of combining partial solutions for the subgraphs induced by its children. Formally, these subgraphs are the following.

**Definition 3.6** (subgraphs induced by a tree decomposition). Let $G = (V, E)$ be a graph and $\mathcal{T}$ be a nice tree decomposition for $G$ with root $r$. Then, for any node $x$ of $\mathcal{T}$, $V_x := \{v \in V \mid v \in B_y$ for a descendant $y$ of $x\}$, and $G_x := (V_x, E_x)$, where $E_x = \{e \in E \mid e$ is introduced in a descendant of $x\}$.

Herein, we consider each node $x$ of $\mathcal{T}$ to be a descendent of itself.

Having defined subgraphs induced by subtrees, we can define partial solutions in them.

### 3.1.2 Partial solutions

Assume that we have a solution path $P$ to VW-SSP. Then, the part of $P$ in $G_x$ is a collection $\mathcal{P}$ of paths (some might consist of a single vertex). When computing a partial solution for a parent $y$ of $x$, we ideally want to check which partial solutions for $x$ can be continued to partial solutions for $y$. However, we cannot try all possible partial solutions for $G_x$—there might be too many. Moreover, this is not necessary: by Definition 3.1(ii)–(iii), vertices in bag $B_y$ cannot be vertices of and cannot have edges to vertices of $V_x \setminus B_x$. Thus, it is enough to know the states of vertices in bag $B_x$ in order to know which partial solutions of $x$ can be continued to $y$. The state of such vertices is characterized by

- which vertices of $B_x$ are end points of paths in $\mathcal{P}$, inner vertices of paths in $\mathcal{P}$, or paths of zero length in $\mathcal{P}$,
- which vertices of $B_x$ are allowed to be neighbors of the solution path $P$,
- how many neighbors the solution path $P$ is allowed to have in $G_x$, and
- which vertices of $B_x$ belong to the same path of $\mathcal{P}$.

We formalize this as follows.

**Definition 3.7** (partial solution). Let $(G, s, t, k, \ell, \kappa, \lambda, \eta)$ be an instance of VW-SSP. For a set $\mathcal{P}$ of paths in $G$ and a set $N$ of vertices in $G$, let

$$\Lambda(\mathcal{P}, N) := \sum_{P \in \mathcal{P}, v \in \mathcal{N}(P)} \lambda(v) + \sum_{P \in \mathcal{P}, v \in \mathcal{V}(P)} \eta(v) + \sum_{v \in N} \lambda(v) \quad \text{and} \quad K(\mathcal{P}) := \sum_{P \in \mathcal{P}, v \in \mathcal{V}(P)} \kappa(v).$$
Moreover, let \( \mathcal{T} \) be a tree decomposition for \( G \), \( x \) be a node of \( \mathcal{T} \), \( D_s \cup D_v \cup D_l \cup N \subseteq B_x \) such that \( \{s, l\} \subseteq D_s \cup D_v \), \( p \) be a partition of \( D := D_s \cup D_v \cup D_l \), and \( l \leq \ell \).

Then, we call \((D_s, D_v, D_l, N, l)\) a pre-signature and \( S = (D_s, D_v, D_l, N, l, p) \) a solution signature at \( x \). A set \( \mathcal{P} \) of paths in \( G_x \) is a partial solution of cost \( K(\mathcal{P}) \) for \( S \) if

i) \( D_z \) are exactly the vertices of zero-length paths \( P \in \mathcal{P} \),

ii) \( D_e \) are exactly the end points of non-zero-length paths \( P \in \mathcal{P} \),

iii) \( D_e \) are exactly those vertices in \( B_x \) that are inner vertices of paths \( P \in \mathcal{P} \),

iv) for each path \( P \in \mathcal{P} \), \( N(P) \cap B_x \subseteq N \),

v) \( \Lambda(\mathcal{P}, N) \leq l \), and

vi) \( \mathcal{P} \) consists of exactly \(|p|\) paths such that each two vertices \( u, v \in D \) belong to the same path of \( \mathcal{P} \) if and only if they are in the same set of the partition \( p \).

For a solution signature \( S \) at a node \( x \), we denote

\[
E_x(S) := \{ \mathcal{P} \mid \mathcal{P} \text{ is a partial solution for } S \},
\]

\[
\min K_x(S) := \min\{ K(\mathcal{P}) \mid \mathcal{P} \in E_x(S) \}.
\]

Obviously, since the root bag \( B_r = \{s, t\} \) by Assumption 3.5, our input instance to VW-SSP is a yes-instance if and only if

\[
\min K_r(\emptyset, \{s, t\}, \emptyset, \ell, \emptyset, \emptyset) \leq k.
\]

(3.1)

Thus, our aim is computing this cost. The naive dynamic programming approach is:

- compute \( \min K_x(S) \) for each solution signature \( S \) and each leaf node \( x \),

- compute \( \min K_x(S) \) for each solution signature \( S \) and each inner node \( x \) under the assumption that \( \min K_x(S') \) has already been computed for all solution signatures \( S' \) at children \( y \) of \( x \).

However, this approach is not suitable to prove Proposition 3.3, since the number of possible solution signatures is too large: the number of different partitions \( p \) of \( tw \) vertices is the \( tw \)-th Bell number, whose best known upper bound is \( O(tw^{tw} \log tw) \). Thus, we do not even have time to look at all solution signatures.

### 3.1.3 Reducing the number of partitions

To reduce the number of needed partitions, we use an approach developed by Bodlaender et al. [10], which also proved its effectiveness in experiments [21]. We will replace the task of computing (3.1) for all possibly partitions by computing only sets of weighted partitions containing the needed information.

**Definition 3.8** (sets of weighted partitions). Let \( \Pi(U) \) be the set of all partitions of \( U \). A set of weighted partitions is a set \( \mathcal{A} \subseteq \Pi(U) \times \mathbb{N} \). For a weighted partition \((p, w) \in \mathcal{A}\), we call \( w \) its weight.

Using sets of weighted partitions, we can reformulate our task of computing \( \min K_x(S) \) for all bags \( B_x \) and all solution signatures \( S \) as follows. Consider a pre-signature \( S = (D_s, D_v, D_e, N, l) \) for a node \( x \) of a tree decomposition. Then, for each \( p \in \Pi(D_s \cup D_v \cup D_e) \), \((S, p)\) is a solution signature. Thus, we can consider

\[
\mathcal{A}_x(S) := \left\{ \left( p, \min_{p \in \Pi(D_s \cup D_v \cup D_e)} K(\mathcal{P}) \right) \mid p \in \Pi(D_s \cup D_v \cup D_e), \quad \mathcal{E}_x(S, p) \neq \emptyset \right\}.
\]

(3.2)

Now, our problem of verifying (3.1) at the root node \( r \) of a tree decomposition is equivalent to checking whether \( \mathcal{A}_r(\emptyset, \{s, t\}, \emptyset, \ell, \emptyset) \) contains a partition \( \{\{s, t\}\} \) of weight at most \( k \). Thus we can, in a classical dynamic programming manner

- compute \( \mathcal{A}_r(S) \) for each pre-signature \( S \) and each leaf node \( x \),
• compute $\mathcal{A}_x(S)$ for each pre-signature $S$ and each inner node $x$ under the assumption that $\mathcal{A}_x(S')$ has already been computed for all pre-signatures $S'$ at children $y$ of $x$.

Yet we will not work with the full sets $\mathcal{A}_x(S)$ but with “representative” subsets of size $2^{\Omega(w)}$. Since the number of pre-signatures is $2^{\Omega(w)} \cdot \ell$, this will allow us to prove Proposition 3.3. In order to formally introduce representative sets of weighted partitions, we need some notation.

**Definition 3.9** (partition lattice). The set $\Pi(U)$ is semi-ordered by the coarsening relation $\subseteq$, where $p \subseteq q$ if every set of $p$ is included in some set of $q$. We also say that $q$ is coarser than $p$ and that $p$ is finer than $q$.

For two partitions $p, q \in \Pi(U)$, by $p \sqcup q$ we denote the (unique) finest partition that is coarser than both $p$ and $q$.

To get an intuition for the $p \sqcup q$ operation, recall from **Definition 3.7** that we will use a partition $p$ to represent connected components of partial solutions: two vertices are connected if and only if they are in the same set of $p$. In these terms, if $p \in \Pi(U)$ are the vertex sets of the connected components of a graph $(U, E)$ and $q \in \Pi(U)$ are the vertex sets of the connected components of a graph $(U, E')$, then $p \sqcup q$ are the vertex sets of the connected components of the graph $(U, E \cup E')$.

Now, assume that there is a solution $P$ to VW-SSP in a graph $G$ and consider an arbitrary node $x$ of a tree decomposition. Then, the subpaths $P'$ of $P$ that lie in $G_x$ are a partial solution for some solution signature $(D_x, D_z, D_{xz}, l, p)$ at $x$. The partition $p$ of $D := D_x \cup D_z \cup D_{xz}$ consists of the sets of vertices of $D$ that are connected by paths in $P'$. Since, in the overall solution $P$, the vertices in $D$ are all connected, the vertices of $D$ are connected in $G \setminus E_x$ according to a partition $q$ of $D$ such that $p \sqcup q = (D)$. Now, if in $P$, we replace the subpaths $P'$ by any other partial solution $P''$ to a solution signature $(D_x, D_z, D_{xz}, l, p')$ such that $K(P'') \leq K(P')$ and $p' \sqcup q = (D)$, then we obtain a solution $P'$ for $G$ with at most the cost of $P$. Thus, one of the two weighted partitions $(p, K(p))$ and $(p', K(p'))$ in $\mathcal{A}_x(D_x, D_z, D_{xz}, l)$ is redundant. This leads to the definition of representative sets of weighted partitions.

**Definition 3.10** (representative sets [10]). For a set $\mathcal{A} \subseteq \Pi(U) \times \mathbb{N}$ of weighted partitions and a partition $q \in \Pi(U)$, let

$$\text{Opt}(q, \mathcal{A}) := \min \{ w \mid (p, w) \in \mathcal{A} \text{ and } p \sqcup q = \{ U \} \}.$$  

Another set $\mathcal{A}' \subseteq \Pi(U) \times \mathbb{N}$ of weighted partitions is said to represent $\mathcal{A}$ if

$$\text{Opt}(p, \mathcal{A}) = \text{Opt}(p, \mathcal{A}') \text{ for all } p \in \Pi(U).$$

A function $f : 2^{\Pi(U) \times \mathbb{N}} \times \mathbb{Z} \to 2^{\Pi(U) \times \mathbb{N}}$ is said to preserve representation if, for all $\mathcal{A}, \mathcal{A}' \subseteq \Pi(U) \times \mathbb{N}$ and all $z \in \mathbb{Z}$, it holds that, if $\mathcal{A}'$ represents $\mathcal{A}$, then $f(\mathcal{A}', z)$ represents $f(\mathcal{A}, z)$. Herein, $\mathbb{Z}$ stands representative for further arguments to $f$.

Transferring this definition to VW-SSP and our sets $\mathcal{A}_x(S)$ in (3.2), $\text{Opt}(q, \mathcal{A}_x(S))$ is the minimum cost of any partial solution for any signature $(S, p)$ at a node $x$ that leads to a connected overall solution when the vertices in $D_x \cup D_z \cup D_{xz}$ are connected in $G \setminus E_x$ as described by partition $q$. Moreover, a subset $\mathcal{A}' \subseteq \mathcal{A}_x(S)$ is representative if this minimum cost for $\mathcal{A}'$ is the same.

**Proposition 3.11** (Bodlaender et al. [10]). Given a set $\mathcal{A} \subseteq \Pi(U) \times \mathbb{N}$ of weighted partitions, a representative subset $\mathcal{A}' \subseteq \mathcal{A}$ with $|\mathcal{A}'| \leq 2^{(\omega - 1)|E|} \cdot |\mathcal{A}| \cdot |U|^{O(1)}$ time, where $\omega < 2.2373$ is the matrix multiplication exponent.

When computing the sets $\mathcal{A}_x(S)$ for pre-signatures $S$ at a node $x$, we will first replace the $A_x(S')$ for all pre-signatures $S'$ at the child nodes $y$ by their representative sets and thus work on sets of size $2^{\Omega(w)}$. More precisely, we will compute $A_x(S)$ from the children sets $A_y(S')$ using the following operators, which Bodlaender et al. [10] have showed to preserve representation.

**Proposition 3.12** (operators on weighted partitions [10]). Let $U$ be a set and $\mathcal{A} \subseteq \Pi(U) \times \mathbb{N}$. The following operations preserve representation.

$$\text{rmc}(\mathcal{A}) := \{ (p, w) \mid \forall (p, w') \in \mathcal{A} : w' \geq w \} \in \Pi(U) \times \mathbb{N}$$

removes duplicate partitions from the set, keeping the one with smallest weight.

$$\mathcal{A} \sqcup \mathcal{B} := \text{rmc}(\mathcal{A} \cup \mathcal{B}) \in \Pi(U) \times \mathbb{N} \text{ for some } \mathcal{B} \subseteq \Pi(U) \times \mathbb{N} \text{ takes all weighted partitions from } \mathcal{A} \text{ and } \mathcal{B},$$

removing copies of larger weight.
We will now show how to use the operators from Proposition 3.12 to compute, for each node \( x \).

Moreover, all operations \( \text{rmc} \) and \( \text{proj} \) are executed in \( O(1) \) time, where \( S \) is the size of the input to the operations, whereas \( \text{join} \) can be executed in \( |A| \cdot |B| \cdot |U|^{O(1)} \) time.

### 3.1.4 The dynamic programming algorithm

We will now show how to use the operators from Proposition 3.12 to compute, for each node \( x \), the weighted set of partitions \( \mathcal{A}_x(S) \) from (3.2) assuming that \( \mathcal{A}_y(S') \) has been computed for all children \( y \) of \( x \) and all pre-signatures \( S' \). Using these operators guarantees that, when applying them to representative subsets, we will again get representative subsets. We describe our algorithm independently for leaf nodes, forget nodes, insert vertex nodes, insert edge nodes, and join nodes.

#### Leaf node \( x \)

By Assumption 3.5, \( B_x = \{ s, t \} \). Moreover, \( G_x \) has no edges, which means that any partial solution in \( G_x \) will contain \( s \) and \( t \) as paths of length zero and will put \( s \) and \( t \) into separate connected components. Thus, for any partition \( D_x \cup D_e \cup D_t \cup N = B_x \), we obtain a partial solution \( \mathcal{A}_x(S) \) according to one of the following cases.

1. **If** \( v \in D_x \cup D_e \cup D_t \), then partial solutions for \( G_x \) exist only if \( \eta(v) \leq l \) and \( v \in D_x \), since the newly introduced vertex has no edges in \( G_x \). Thus, we add \( v \) as a zero-length path to all partial solutions with \( A(P, N) \leq l - \eta(v) \) for \( G_x \) and increase their cost by \( \kappa(v) \).

2. **If** \( v \in N \), then partial solutions for \( G_x \) exist only if \( l \geq \lambda(v) \). Then, we obtain a partial solution for \( G_x \) from each partial solution \( \mathcal{P} \) for \( G_v \) with \( A(P, N \setminus \{ v \}) \leq l - \lambda(v) \).

3. **If** neither of the above, then any partial solution for \( G_x \) is also one for \( G_v \) of the same cost.

Thus,

\[
\mathcal{A}_x(S) = \begin{cases} 
\text{shift}(\kappa(v), \text{ins}(\{v\}, \mathcal{A}_y(D_x \cup \{ v \}, D_e, D_t, N, l - \eta(v)))) & \text{if } v \in D_x, \\
\mathcal{A}_x(D_x, D_e, D_t, N \setminus \{ v \}, l - \lambda(v)) & \text{if } v \in N \text{ and } l \geq \lambda(v), \\
\mathcal{A}_x(D_x, D_e, D_t, N, l) & \text{if } v \notin D_x \cup D_e \cup D_t \cup N, \\
\end{cases}
\]

\[3\] Bodlaender et al. [10] actually require \( w^t \in \mathbb{N} \) here. Yet from the proof of their Lemma 3.6, it is easy to see that shift preserves representation whenever it returns a set of partitions with nonnegative weights.
Introduce edge \([u,v]\) node \(x\) with child \(y\). For any partition \(D_x \uplus D_y \uplus D_t \uplus N \subseteq B_x\) with \([s,t] \subseteq D_y \uplus D_t, l \in \mathbb{N}\), and pre-signature \(S = (D_x, D_y, D_t, N, l)\), we compute \(\mathcal{A}_y(S)\) according to one of three cases for \(u\) and \(v\):

(IE1) If \([u,v] \cap N \neq \emptyset\), then \([u,v] \cap (D_x \uplus D_y \uplus D_t \uplus N) = \emptyset\), or if \([u,v] \subseteq D_x \uplus D_y \uplus D_t\) and \([u,v] \cap D_y \neq \emptyset\), then edge \([u,v]\) cannot be part of a path in a partial solution to \(G_s\). Moreover, it is either not incident to a path vertex or one of its endpoints is already in \(N\) and thus has been accounted for. Thus, any partial solution for \(G_s\) satisfying \(S\) is a partial solution of the same cost for \(G_z\).

(IE2) For \([u,v] \subseteq D_x \uplus D_t\) then we have two choices: take edge \([u,v]\) into a path of a partial solution or not. If we take it, then the connected components containing \(u\) and \(v\) in partial solutions to \(G_s\) will be one connected component in any partial solution to \(G_z\). Moreover, if we add edge \([u,v]\) to a solution path and \(u \in D_x\), then \(u\) must be a path of zero length in the partial solution to \(G_s\) and if \(u \in D_t\), then \(u\) must be an end point of a path in the partial solution to \(G_z\). Symmetrically, this holds for \(v\).

(IE3) Neither of both. Since (IE1) does not apply, we have \([u,v] \cap N = \emptyset\) and, modulo symmetry, \(v \in D_x \uplus D_y \uplus D_t\) and \(u \notin D_x\). Moreover, if \(v \in D_t\), then \(u \notin D_x \uplus D_t\). If \(v \in D_t\), then we also get \(u \notin D_x \uplus D_t\) because (IE2) does not apply. Hence, \(v\) is required to be part of a path in our partial solution to \(G_s\), whereas its neighbor \(u\) is not allowed to be on any path nor neighbor of a path. There is no such feasible solution.

Thus, \(\mathcal{A}_y(S) = \)

\[
\begin{cases}
\mathcal{A}_y(S) & \text{if } [u,v] \cap N \neq \emptyset, \\
\mathcal{A}_y(S) & \text{if } [u,v] \cap (D_x \uplus D_y \uplus D_t) = \emptyset, \\
\mathcal{A}_y(S) & \text{if } [u,v] \subseteq D_x \uplus D_y \uplus D_t \cup N \\
\mathcal{A}_y(S) \uplus \text{glue}([u,v], \mathcal{A}_y(S[[[u,v]]_1 \to D_x])) & \text{if } [u,v] \subseteq D_x, \\
\mathcal{A}_y(S) \uplus \text{glue}([u,v], \mathcal{A}_y(S[[[u,v]]_1 \to D_t])) & \text{if } [u,v] \subseteq D_t, \\
\mathcal{A}_y(S) \uplus \text{glue}([u,v], \mathcal{A}_y(S[[[u,v]]_1 \to D_y])) & \text{if } v \in D_y, u \in D_x, \\
\mathcal{A}_y(S) \uplus \text{glue}([u,v], \mathcal{A}_y(S[[[v]]_1 \to D_y, [u]]_1 \to D_x)) & \text{if } u \in D_y, v \in D_x, \\
\emptyset & \text{otherwise,}
\end{cases}
\]

where

\[
S[[[u,v]]_1 \to D_x] := (D_x \uplus \{u,v\}, D_x \setminus \{u,v\}, D_t, N, l),
\]

\[
S[[[u,v]]_1 \to D_t] := (D_t \uplus \{u,v\}, D_t \setminus \{u,v\}, D_x, N, l),
\]

\[
S[[[u,v]]_1 \to D_y] := (D_y \uplus \{u\}, D_y \setminus \{u\}, D_x \cup \{v\}, N, l).
\]

Forget vertex \(v\) node \(x\) with child \(y\). By Assumption 3.5, the forgotten vertex \(v \notin [s,t]\). To extend partial solutions for \(G_s\) to \(G_z\), we consider three possibilities of which role the forgotten vertex \(v\) could have played in a partial solution for \(G_s\).

(F1) \(v\) could have been a neighbor. In this case, we already accounted for the cost \(\lambda(v)\) when \(v\) was introduced. Moreover, by Definition 3.1(ii) and (iii), no parent node of \(x\) will ever introduce an edge incident to \(v\). Thus, we can safely ignore \(v\) in \(x\) and all parent nodes.

(F2) \(v\) could have been part of a path. In this case, \(v\) must be an inner vertex of any partial solution for \(G_s\), since \(v\) must be an inner vertex of the overall solution and no parent node of \(x\) will be able to introduce edges incident to \(v\) due to Definition 3.1(ii) and (iii). Thus, we simply remove \(v\) from all partitions in signatures for partial solutions of \(G_s\) that contain \(v\) as an inner vertex. This will not reduce the number of sets in these partitions, since the end vertices of the path containing \(v\) in a partial solution for \(G_s\) are both connected to \(v\) and in \(B_y\).

(F3) Neither of both. In this case, any partial solution for \(G_s\) that does not contain \(v\) as neighbor nor path vertex is a partial solution of the same cost for \(G_z\).

Thus, for any partition \(D_x \uplus D_y \uplus D_t \uplus N \subseteq B_x\) with \([s,t] \subseteq D_y \uplus D_t, l \in \mathbb{N}\), and pre-signature \(S = (D_x, D_y, D_t, N, l)\),

\[
\mathcal{A}_y(S) = \mathcal{A}_y(D_x, D_y, D_t, N \cup \{v\}, l) \uplus \text{proj}(v, \mathcal{A}_y(D_x, D_y, D_t \uplus \{v\}, N, l)) \uplus \mathcal{A}_y(D_x, D_y, D_t, N, l).
\]
Join node with children \(y\) and \(z\). For any partition \(D_x \cup D_z \cup D_y \cup D_z \cup N \subseteq B_i\) with \([s, t]\) \(\subseteq D_x \cup D_z\), \(l \in \mathbb{N}\), and pre-signature \(S = (D_x, D_z, D_y, N, l_i)\), to compute \(\mathcal{A}_i(S)\), we consider the roles of each vertex \(v \in B_i\).

(J1) If \(v \in N\), then our partial solution in \(G_x\) allows \(v\) as a neighbor. The partial solutions for \(G_x\) and \(G_z\) that we seek to combine thus also must allow \(v\) as a neighbor.

(J2) If \(v \in D_x\), then \(v\) must be a path of length zero also in the partial solutions to \(G_y\) and \(G_z\).

(J3) If \(v \in D_z\), then there are two choices: \(v\) is a path of length zero in a partial solution to \(G_z\) and an end point of a path of non-zero length in a partial solution to \(G_x\), or vice versa.

(J4) If \(v \in D_y\), then there are three choices: \(v\) might be the end point of a path of non-zero length in the partial solutions to each of \(G_x\) and \(G_z\), or it might be an inner vertex in a partial solution to \(G_x\) and a path of length zero in a partial solution to \(G_z\), or vice versa.

(J5) Otherwise, \(v\) is not allowed to be part of a partial solution to \(G_x\) nor allowed to neighbor it. Thus, \(v\) is also disallowed to be part of partial solutions or to neighbor them in \(G_y\) and \(G_z\).

To compute the \(\mathcal{A}_i(S)\), let
\[
D := D_x \cup D_z \cup D_y, \quad \lambda_N := \sum_{v \in N} \lambda(v) + \sum_{v \in D} \eta(v), \quad \kappa_D := \sum_{v \in D} \kappa(v).
\]

Any partial solution \(\mathcal{P}\) for \(G_x\) with \(\Lambda(\mathcal{P}, N) \leq l\) decomposes into a partial solution \(\mathcal{P}_y\) for \(G_y\) and a partial solution \(\mathcal{P}_z\) for \(G_z\). By Definition 3.1(ii) and (iii), the set of vertices common to \(\mathcal{P}_y\) and \(\mathcal{P}_z\) lies in \(B_i\) and, hence, is precisely \(D\). Thus
\[
K(\mathcal{P}_y) + K(\mathcal{P}_z) - \kappa_D = K(\mathcal{P}) \geq 0.
\]

Moreover, all common neighbors of \(\mathcal{P}_y\) and \(\mathcal{P}_z\) lie in \(N\). Thus, \(\Lambda(\mathcal{P}_y, N) \leq l_y\) and \(\Lambda(\mathcal{P}_z, N) \leq l_z\) for some
\[
l_y + l_z = l + \lambda_N.
\]

Thus, by (J1)–(J5), one has
\[
\mathcal{A}_i(S) := \bigcup_{D_x \subseteq D} \text{shift}\{\kappa_D, \text{join}(\mathcal{A}_i(D_x^f, D_x^l, N, l), \mathcal{A}_i(D_z^f, D_z^l, N, l))\}.
\]

(3.4)

where the \(D_x^f, D_x^l\) and \(D_y^l\) are fully determined by the choice of \(D_x^l, D_y^l\) and \(D_y^l\) via
\[
D_x^f = D_x \cup D_y \cup (D_z \cap D_x^l), \quad D_x^l = (D_x \cap D_y^l) \cup (D_z \cap D_x^l), \quad D_z^l = D \setminus (D_x^l \cup D_y^l).
\]

Wrapping up. Having described how to compute (3.2) for each node type of a nice tree decomposition, we are now ready to prove Proposition 3.3 exploiting that we can efficiently compute small representative subsets of our families of weighted partitions using Proposition 3.11. We will apply this shrinking procedure to all intermediate sets computed in our algorithm.

Proof of Proposition 3.3. Our algorithm works as follows. It first preprocesses the given tree decomposition according to Assumption 3.5, which can be done in \(n \cdot \text{tw}(1)\) time and thus gives a tree decomposition with \(n \cdot \text{tw}(1)\) bags [10]. Henceforth, we will be working on a tree decomposition of width at most \(\text{tw} + 2\), that is, each bag has size at most \(\text{tw} + 3\).

The algorithm now computes (3.2) for each node of the tree decomposition and each pre-signature \(S\) as described in Sections 3.1.4 to 3.1.4. However, after computing \(\mathcal{A}_i(S)\) for some pre-signature \(S\) at some node \(x\), it will use Proposition 3.11 to store only a representative subset \(\mathcal{A}_i(S)\) with \(|\mathcal{A}_i(S)| \leq 2^{\text{tw} + 3} - 1\). Since we compute \(\mathcal{A}_i(S)\) only using operators in Proposition 3.12, the set \(\mathcal{A}_i(S)\) represents \(\mathcal{A}_i(S)\) at each node \(x\) of the tree decomposition, in particular at the root node \(r\), where we can now verify whether \(\mathcal{A}_i(\emptyset, \emptyset, \emptyset, \emptyset, \emptyset)\) contains a partition \([\{s, t\}]\) of weight at most \(k\).
We analyze the running time of this algorithm. For each pre-signature \( S \), each leaf node can be processed according to Section 3.1.4 in constant time and the stored representative subset \( \mathcal{R}(S) \subseteq \mathcal{R}(S) \) has constant size and can be computed in constant time. According to Definition 3.7, there are at most \( 5^{tw+1} \cdot \ell \) pre-signatures, since the bags of our tree decomposition have size at most \( tw + 3 \). Thus, each leaf node can be processed in \( O(5^{tw} \cdot \ell) \) time.

For each pre-signature \( S \) with \( |D_x \cup D_e \cup D_l| = i \), the most expensive operation when processing introduce vertex, introduce edge, and forget nodes is the union operation in (3.3), which is applied to three sets of weighted partitions, each of a size upper bounded by \( 2^{tw+1} - 1 \). By Proposition 3.12, this union can be computed in \( 3 \cdot 2^{tw+1} \cdot r^{O(1)} = 2^\ell \cdot r^{O(1)} \) time. The resulting set of weighted partitions therefore has size at most \( 2^\ell \cdot r^{O(1)} \). Thus, shrinking using Proposition 3.11 works in \( 2^{(\omega - 1)\ell} \cdot 2^\ell \cdot r^{O(1)} \) time, which is the most expensive operation for each fixed pre-signature. Since there are at most \( \binom{tw+3}{3} \cdot 3^i \cdot 2^{tw+3-i} \cdot \ell \) pre-signatures with \( |D_x \cup D_e \cup D_l| = i \), each introduce vertex, introduce edge, and forget node is processed in

\[
\sum_{i=1}^{tw+3} \binom{tw+3}{i} \cdot 3^i \cdot 2^{tw+3-i} \cdot \ell \cdot 2^\omega \cdot r^{O(1)} = \ell \cdot tw^{O(1)} \cdot \sum_{i=1}^{tw+3} \binom{tw+3}{i} \cdot (3 \cdot 2^\omega)^i \cdot 2^{tw+3-i} = \ell \cdot tw^{O(1)} \cdot (2 + 3 \cdot 2^\omega)^{tw+3} \text{ time.}
\]

For each pre-signature \( (D_x, D_e, D_l, N, I) \) with \( |D_x| = i_x, |D_e| = i_e, |D_l| = i_l \), and \( i_x + i_e + i_l = i \), processing a join node according to Section 3.1.4 is more costly. By (1)–(35), the union operator in (3.4) has up to \( 1^i \cdot 2^i \cdot 3^i \cdot I \) operands. Each operand is a join of two sets of size at most \( 2^i - 1 \) and, by Proposition 3.12, takes \( 4^i \cdot tw^{O(1)} \) time to compute. Thus, the union operator is applied to sets whose total size is bounded by \( 2^i \cdot 3^i \cdot 4^i \cdot I \). Shrinking according to Proposition 3.11 thus works in \( 2^{(\omega - 1)\ell} \cdot 2^i \cdot 3^i \cdot 4^i \cdot I \cdot tw^{O(1)} \) time and, again, is the most expensive operation in the computation of a join node for a fixed pre-signature. There are

\[
\binom{tw+3}{i} \cdot \sum_{i_x, i_e, i_l} \binom{i}{i_x, i_e, i_l} \cdot 2^{tw+3-i} \cdot \ell \cdot 2^\omega \cdot r^{O(1)}
\]

pre-signatures \( (D_x, D_e, D_l, N, I) \) with \( |D_x| = i_x, |D_e| = i_e, |D_l| = i_l \), and \( i_x + i_e + i_l = i \), where \( \binom{i}{i_x, i_e, i_l} \) is the multinomial coefficient—the number of ways to throw \( i \) items into three distinct bins such that the first bin gets \( i_x \) items, the second gets \( i_e \) items, and the third gets \( i_l \) items. Thus, each join node is processed in a total time of

\[
\sum_{i=1}^{tw+3} \binom{tw+3}{i} \cdot \sum_{i_x, i_e, i_l \equiv i} \binom{i}{i_x, i_e, i_l} \cdot 2^{tw+3-i} \cdot \ell \cdot 2^\omega \cdot r^{O(1)} = \ell^2 \cdot tw^{O(1)} \cdot \sum_{i=1}^{tw+3} \binom{tw+3}{i} \cdot \sum_{i_x, i_e, i_l \equiv i} \binom{i}{i_x, i_e, i_l} \cdot 1 \cdot 2^i \cdot 3^i \cdot 2^{tw+3-i} \cdot 2^\omega \cdot r^{O(1)}
\]

which, by the multinomial theorem, is

\[
\ell^2 \cdot tw^{O(1)} \cdot \sum_{i=1}^{tw+3} \binom{tw+3}{i} \cdot 6^i \cdot 2^{tw+3-i} \cdot 2^i \cdot 2^\omega
\]

\[
\ell^2 \cdot tw^{O(1)} \cdot \sum_{i=1}^{tw+3} \binom{tw+3}{i} \cdot (12 \cdot 2^\omega)^i \cdot 2^{tw+3-i} = l \cdot tw^{O(1)} \cdot (2 + 12 \cdot 2^\omega)^{tw+3}.
\]

Since our tree decomposition has \( n \cdot tw^{O(1)} \) bags, we conclude that we can solve VW-SSP in

\[
(n \cdot \ell^2 \cdot tw^{O(1)} \cdot (2 + 12 \cdot 2^\omega)^{tw+3}) \text{ time.}
\]

\[\square\]

3.2 Limits of data reduction

In the previous section, we presented an algorithm that efficiently solves SSP in tree-like graphs, namely, in graphs of small treewidth. In the following, we study the possibilities for provably effective polynomial-time data reduction in tree-like graphs.

In this section, we continue studying the treewidth parameter. Yet we prove that, unless \( \text{coNP} \subseteq \text{NP/poly} \), SSP has no problem kernel of size polynomial in the treewidth of the input graph. In fact, we prove the following, even stronger theorem:
Theorem 3.13. **Short Secluded Path has no problem kernel with size polynomial in tw + k + ℓ**, even on planar graphs with maximum degree six, where tw is the treewidth, unless coNP \(\subseteq NP/poly\) and the polynomial-time hierarchy collapses to the third level.

To prove Theorem 3.13, we use a special kind of reduction called cross composition [9].

Definition 3.14 (cross composition). A polynomial equivalence relation \(~\) is an equivalence relation over \(\Sigma^*\) such that

- there is an algorithm that decides \(x \sim y\) in polynomial time for any two instances \(x, y \in \Sigma^*\), and such that
- the index of \(~\) over any finite set \(S \subseteq \Sigma^*\) is polynomial in \(\max_{x \in S} |x|\).

A language \(K \subseteq \Sigma^*\) cross-composes into a parameterized language \(L \subseteq \Sigma^* \times \mathbb{N}\) if there is a polynomial-time algorithm, called cross composition, that, given a sequence \(x_1, \ldots, x_p\) of \(p\) instances that are equivalent under some polynomial equivalence relation, outputs an instance \((x^*, k)\) such that

- \(k\) is bounded by a polynomial in \(\max_{i=1}^p |x_i| + \log p\) and
- \((x^*, k) \in L\) if and only if there is an \(i \in \{1, \ldots, p\}\) such that \(x_i \in K\).

Cross compositions can be used to rule out problem kernels of polynomial size using the following result:

Proposition 3.15 (Bodlaender et al. [9]). If a NP-hard language \(K \subseteq \Sigma^*\) cross-composes into the parameterized language \(L \subseteq \Sigma^* \times \mathbb{N}\), then there is no polynomial-size problem kernel for \(L\) unless coNP \(\subseteq NP/poly\) and the polynomial-time hierarchy collapses to the third level.

Using a cross composition, Luckow and Fluschnik [41] proved that SSP on planar graphs of maximum degree six does not admit a problem kernel with size polynomial in \(k + \ell\). To prove Theorem 3.13, we show that the graph created by their cross composition has treewidth at most \(3n + 3\), where \(n\) is the number of vertices in each input instance to their cross composition. To this end, we briefly describe their composition.

Construction 3.16. For \(i \in \{1, \ldots, p\}\), let \((G_i, s_i, t_i, k_i, \ell_i)\) be instances of SSP such that each \(G_i\) is a planar graph of maximum degree five and has a planar embedding with \(s_i\) and \(t_i\) on the outer face. Without loss of generality, \(p\) is a power of two (otherwise, we pad the list of input instances with no-instances), the vertex sets of the graphs \(G_1, \ldots, G_p\) are pairwise disjoint, and, for all \(i \in \{1, \ldots, p\}\), one has \(V(G_i) = n, \ell_i = \ell, \) and \(k_i = k\) (this is a polynomial equivalence relation). We construct an instance \((G, s, t, k', \ell')\) of SSP, where

\[
k' := 3k + 2 \log p, \quad \ell' := \ell + 2 \log p - 1,
\]

and the graph \(G\) is as follows. Graph \(G\) consists of \(G_1, \ldots, G_p\) and two rooted balanced binary trees \(T_s\) and \(T_t\), with roots \(s\) and \(t\), respectively, each having \(p\) leaves. Let \(g_1, \ldots, g_{2p-1}\) and \(h_1, \ldots, h_{2p-1}\) denote the vertices of \(T_s\) and \(T_t\) enumerated by a depth-first search starting at \(s\) and \(t\), respectively. Moreover, let \(a_1, \ldots, a_p\) and \(b_1, \ldots, b_p\) denote the leaves of \(T_s\) and \(T_t\) as enumerated in each depth-first search mentioned before. Then, for each \(i \in \{1, \ldots, p\}\), graph \(G\) contains an \(a_i\)-\(s_i\)-path and a \(b_i\)-\(t_i\)-path, each on its own set of \(k\) inner vertices.

Proof of Theorem 3.13. Luckow and Fluschnik [41] already proved that Construction 3.16 is a correct cross composition. Moreover, obviously, \(k', \ell' \in O(n + \log p)\). Thus, to prove Theorem 3.13, it remains to prove \(tw(G) \leq 3n + 3\) for the graph \(G\) constructed by Construction 3.16. To this end, we give a tree decomposition of width at most \(3n + 3\) for \(G\) (recall Definition 3.1 of tree decompositions). The construction of the tree decomposition is illustrated in Figure 3.2.

First, we construct a tree decomposition \(T_s = (T_s, \beta_s)\) of \(T_s\) with bags as follows. Let \(parent_{T_s}(v)\) denote the parent of \(v \in V(T_s)\) (where \(parent_{T_s}(s) = s\)). For each \(v \in V(T_s)\), let \(\beta_s(v) := \{v, parent_{T_s}(v)\}\). Then \(T_s\) is a tree decomposition of width one.

Now, let \(T_t = (T_t, \beta_t)\) be the tree decomposition for \(T_t\) constructed analogously. We construct a tree decomposition \(T_{st} = (T_{st}, \beta_{st})\) for the disjoint union of \(T_s\) and \(T_t\) as follows: take \(T = T_s \cup T_t\), and, for each \(i \in \{1, \ldots, 2p - 1\}\), let \(\beta_{g_i} := \beta_s(g_i) \cup \beta_t(h_i)\), where \(g_i\) and \(h_i\) are the vertices of \(T_s\) and \(T_t\) according to the depth-first labeling in Construction 3.16. As \(T_s\) and \(T_t\) are tree decompositions of two vertex-disjoint trees \(T_s\) and \(T_t\), respectively, and \(\{g_i, g_j\}\) is an edge of \(T_t\) if and only if \(\{h_i, h_j\}\) is an edge of \(T_s\), \(T_{st}\) is a tree decomposition for the disjoint union of \(T_s\) and \(T_t\). The width of \(T_{st}\) is three.
Now, recall that, for \( i \in \{1, \ldots, p\} \), the graph \( G_i \) in \( G \) is adjacent to exactly one leaf \( a_i \) of \( T_s \) and one leaf \( b_i \) of \( T_t \) (via paths on \( k \) vertices each). Hence, we obtain a tree decomposition \( \mathcal{T} \) of \( G \) from \( \mathcal{T}_i \) by, for each \( i \in \{1, \ldots, p\} \), adding \( V(G_i) \) and all the vertices in the path connecting \( a_i \) with \( s_i \) and in the path connecting \( t_i \) with \( b_i \) to bag \( \beta(a_i) \), which contains both \( a_i \) and \( b_i \). The width of \( \mathcal{T} \) is at most \( n + 2k + 3 \leq 3n + 3 \), and hence, we have \( \text{tw}(G) \leq 3n + 3 \). □

### 4 Graphs will small feedback sets

In the previous section, we have shown that SSP has no problem kernel with size polynomial in the treewidth of the input graph. In this section, we complement this result by studying polynomial-size kernelizability of SSP with respect to other parameters that measure the tree-likeness of a graph: the feedback vertex number \( f_{\text{v}} \) and feedback edge number \( f_{\text{e}} \).

In Section 4.1, we prove problem kernels with size polynomial in \( f_{\text{v}} \) and polynomial in \( f_{\text{e}} + k + \ell \). In Section 4.2, we show that SSP does not allow for problem kernels of size polynomial in \( f_{\text{v}} + \ell \) unless the polynomial-time hierarchy collapses.

#### 4.1 Efficient data reduction

In this section, we prove problem kernels with size polynomial in \( f_{\text{v}} \) and polynomial in \( f_{\text{e}} + k + \ell \). In both kernelizations, we will define a notion of good vertices and then use the following notation.

**Definition 4.1.** We call an \( a-b \)-path in a tree \( \mathcal{T} \) edgy if it contains no good vertex and no vertex \( w \) with \( \text{deg}_\mathcal{T}(w) \geq 3 \). We call an \( a-b \)-path \( Q \) maximal-edgy if there is no edgy path containing \( Q \) with more vertices than \( Q \).

Our data reduction rules will will delete leaves from trees and shrink their maximal-edgy paths, storing information about the changes in vertex weights. That is, the result of our data reduction will be an instance of \textsc{Vertex-Weighted Short Secluded Path}. These instances, which we will call simple, satisfy a set of properties that allow us to strip them of weights efficiently.

**Definition 4.2.** An instance \((G, s, t, k, \ell, \lambda, \kappa, \eta)\) of \textsc{VW-SSP} with \( G = (V, E) \) is called simple if there is a set \( A \subseteq V \) such that

1. \( \kappa(s) = \kappa(t) = 1 \),
2. \( \lambda(v) = 1 \) for all \( v \in V \),
3. \( \eta(v) > \ell \) and \( \kappa(v) = 1 \) for all \( v \in A \), and

---

### Figure 3.2: Overview on the tree decompositions (sets in boxes refer to the bags), exemplified for \( p = 4 \) input instances. (a) and (b) display the tree decomposition for \( T_s \) and \( T_t \), respectively. (c) displays the tree decomposition \( \mathcal{T} \) (and \( \mathcal{T}_i \) when removing \( V_i \) for \( i \in \{1, \ldots, 4\} \)). Here, \( V_i \) represents the set of vertices in the input graph \( G_i \) together with the vertices in the paths connecting \( G_i \)'s source and sink with the leaves on the binary trees.
To prove Proposition 4.3, we use the following construction.

**Proposition 4.3.** Any simple instance $(G, s, t, k, ℓ, λ, κ, η)$ of VW-SSP with $G = (V, E)$ and given $A \subseteq V$ can be reduced to an equivalent instance of SSP with at most $M$ vertices in time linear in $M := \kappa(V) + η(V)$.

To prove Proposition 4.3, we use the following construction.

**Construction 4.4.** Let $(G, s, t, k, ℓ, λ, κ, η)$ be a simple instance of VW-SSP with $G = (V, E)$ and given set $A \subseteq V$ as in **Definition 4.2**. Construct an instance $(G', s', t', k, ℓ)$ of SSP as follows. Let $G'$ be initially a copy of $G$. For each $v \in V$ with $κ(v) > 1$, let $\{v', v''\} = N_{G-A}(v)$, replace $v$ by a path $P_v$ with $κ(v)$ vertices, make one endpoint adjacent to $v'$, and the other endpoint adjacent to $v''$. Next, for each $v \in V$, add a set $U_v$ of $η(v)$ vertices. If $κ(v) = 1$ make each $u \in U_v$ only adjacent to $v$. If $κ(v) > 1$ make each $u \in U_v$ only adjacent to some vertex $x \in P_v$. Finally, for each $v \in V \setminus (A \cup \{s, t\})$ with $κ(v) > 1$ and $W := N_G(v) \cap A \neq \emptyset$, make each $w \in W$ adjacent with some vertex on $P_v$. This finishes the construction of $G'$. Observe that the construction can be done in time linear in $M$ and $(G', s, t, k, ℓ)$ consists of $M$ vertices.

**Proof of Proposition 4.3.** Let $(G, s, t, k, ℓ, λ, κ, η)$ be a simple instance of VW-SSP with $G = (V, E)$ and given set $A \subseteq V$ as in **Definition 4.2**. Apply Construction 4.4 to compute instance $I' = (G', s, t, k, ℓ)$ of SSP with at most $M$ vertices in time linear in $M := \kappa(V) + η(V)$. We claim that $I$ is a yes-instance if and only if $I'$ is a yes-instance.

$(⇒)$ Let $I$ be a yes-instance and $P := (v_1, v_2, \ldots, v_q)$ with $v_1 = s$ and $v_q = t$ be a solution $s$-$t$-path. Let $W \subseteq V(P)$ denote the vertices in $P$ with $κ(v) > 1$. We claim that the path $P'$ obtained from $P$ by replacing each vertex $v \in W$ by $P_v$ is a solution $s$-$t$-path to $I'$. First, observe that $|V(P')| = |V(P)| + |κ(W)| \leq k$. It remains to prove (recall that $κ(v) = 1$ for all $v \in V$)

$$|N_G(V(P'))| = |N_G(V(P))| + \sum_{v \in V(P)} |U_v| = |N_G(V(P))| + \sum_{v \in V(P)} η(v) \leq ℓ.$$  

To this end, it is enough to prove $N_G(V(P')) = N_G(V(P)) \cup \bigcup_{v \in V(P)} U_v$. First observe that no vertex in $A$ is contained in $V(P)$ since $|U_v| > ℓ$ for all $v \in V$. Moreover, no vertex in $A$ is in $V(P)$ since $η(v) > ℓ$ for all $v \in A$. Since $I$ is simple, if $V(P)$ contains no vertex $v$ with $κ(v) > 1$, that is $W = \emptyset$, then it contains no vertex $w$ neighboring a vertex $v$ with $κ(v) > 1$ since $w$ is of degree two in $G-A$, distinct from $s$ and $t$, and no vertex in $A$ is contained in $V(P)$. Hence, if $W = \emptyset$, then $N_G(V(P')) = N_G(V(P)) \cup \bigcup_{v \in V(P)} U_v$. Thus, assume $W = \{v_1, \ldots, v_q\}$ with $q \geq 1$ and let $v'$ and $v''$ be the only two neighbors of $v_i$ in $G-A$ for all $i \in \{1, \ldots, q\}$. Then, for each $v \in W$, we have $N_G(V(P_v)) \setminus \{v', v''\} = N_G(V(P_v)) \setminus \{v', v''\}$, since the neighbors of $v$ in $A$ coincide with the neighbors of $V(P_v)$ in $A$. Thus,

$$N_G(V(P')) = \left( N_G(V(P)) \setminus W \right) \bigcup_{1 \leq i \leq q} V(P_v) \bigcup \left( N_G(V(P_v)) \setminus \{v', v''\} \right) \cup \bigcup_{v \in V(P)} U_v$$

$$= (N_G(V(P)) \setminus W) \cup \bigcup_{1 \leq i \leq q} (N_G(V(P_v)) \setminus \{v', v''\}) \cup \bigcup_{v \in V(P)} U_v$$

$$= N_G(V(P)) \cup \bigcup_{v \in V(P)} U_v.$$

$(⇐)$ Let $I'$ be a yes-instance and let $P'$ be a solution $s$-$t$-path. Note that all vertices in $P_v$ for $v \in V$ with $κ(v) > 1$ are of degree two and distinct from $s$ and $t$. Hence, if a vertex of $P_v$ is contained in $P'$, then all vertices from $P_v$ are contained in $P'$. Let $W \subseteq V$ denote the set of vertices $v$ with $κ(v) > 1$ such that $P_v$ is a subpath of $P'$. We claim that the path $P$ obtained from $P'$ by replacing each path $P_v$ by a $v$ in $W$ is a solution $s$-$t$-path for $I$. First, observe that $κ(V(P)) = |V(P')| - \sum_{v \in W} |V(P_v)| + |κ(W)| = |V(P')| \leq k$. Second, similar as in the converse direction, since $I$ is simple, we have

$$|N_G(V(P))| + \sum_{v \in V(P)} η(v) = |N_G(V(P))| + \sum_{v \in V(P)} |U_v| = |N_G(V(P'))| \leq ℓ.$$  

\[\ □\]
4.1.1 A problem kernel with $O(fes)$ vertices and edges

We show two data reduction algorithms that reduce SSP to an equivalent instance of VW-SSP with $O(fes)$ vertices and allow a trade-off between the running time and the size of the resulting instance. The first runs in linear time, yet creates vertices with weights in $O(k + \ell)$, thus not bounding the overall size of the reduced instance by a polynomial in $fes$. The second takes polynomial time and creates vertex weights encodable using $O(fes^2)$ bits. Thus, when finally reducing back to SSP using Proposition 4.3, we obtain a problem kernel of size $O(fes \cdot (k + \ell))$ using the first algorithm and a problem kernel of size $O(fes^2)$ using the second algorithm.

Theorem 4.5. Short secluded path admits a problem kernel

(i) with size $O(fes \cdot (k + \ell))$ that is computable in linear-time, and

(ii) with size polynomial in $fes$ that is computable in polynomial time.

Herein, $fes$ denotes the feedback edge number of the input graph.

We first prove (i). Concretely, we prove the following:

Proposition 4.6. For any instance of SSP we can compute in linear time an equivalent simple instance of VW-SSP with $16fes + 9$ vertices, $17fes + 8$ edges and vertex weights in $O(k + \ell)$.

Let $F$ be a feedback edge set of size $fes$ in $G = (V, E)$. By Reduction Rule 1.4, we may assume $G$ to be connected. Thus, $T := G - F$ is a tree. Let $Y := \{v \in V | v \in E \} \cup \{s, t\}$ denote the set of vertices containing $s$ and $t$ and all endpoints of the edges in $F$. We call the vertices in $Y$ good. In the following, we will interpret the input SSP instance as an instance of VW-SSP with vertex weights $\lambda : V \rightarrow \mathbb{N} \cup \{0\}$, $\nu \mapsto 1$, $\kappa : V \rightarrow \mathbb{N}$, $\nu \mapsto 1$, and $\eta : V \rightarrow \mathbb{N} \cup \{0\}, \nu \mapsto 0$. Our first data reduction rule deletes vertices and edges.

Reduction Rule 4.7. For each $v \in V(T) \setminus Y$ with $N_T(v) = \{w\}$, set $\eta(w) := \min\{\ell + 1, \eta(w) + 1\}$ and delete $v$.

Next, we shrink any long degree-two edge path to a path with three vertices, storing all information about the path in the weights of the vertices of the new path.

Reduction Rule 4.8. Let $Q \subseteq T$ be a maximal-edgy $a$-$b$-path with $|V(Q)| > 3$ in $T$ and let $K := V(Q) \setminus \{a, b\}$. Then, add a vertex $x$ and the edges $\{x, a\}$ and $\{x, b\}$. Set $\kappa(x) := \min\{k + 1, \kappa(K)\}$ and $\eta(x) := \min\{\ell + 1, \eta(K)\}$. Delete all vertices in $K \setminus \{a, b\}$.

After applying Reduction Rules 4.7 and 4.8 exhaustively, we can show:

Observation 4.9. Let $T$ be such that none of Reduction Rules 4.7 and 4.8 is applicable. Then $G$ has at most $8|Y| - 7$ vertices and $8|Y| - 8 + |F|$ edges, where each vertex is of weight $O(k + \ell)$.

Proof. Due to Reduction Rule 4.7, every leaf of $T$ is in $Y$. Hence, there are at most $2|Y| - 1$ vertices in $T$ of degree less than three. Due to Reduction Rule 4.8, these paths contain at most three vertices. It follows that there are at most $8|Y| - 7$ vertices in $T$, each of weight $O(k + \ell)$, and, consequently, at most $8|Y| - 8$ edges in $T$. As $T$ only differs from $G$ by $F$, it follows that $G$ has at most $8|Y| - 8 + |F|$ edges.

We are ready to prove Proposition 4.6.

Proof of Proposition 4.6. Let $I = (G, s, t, k, \ell)$ be an instance of SSP. Compute a minimum feedback vertex set $F$ of size $fes := |F|$ in $G$ in linear time (just take the complement of a spanning tree). Compute the set $Y$ of good vertices. First apply Reduction Rule 4.7 exhaustively in linear time. Next, apply Reduction Rule 4.8 exhaustively in linear time. Let $I' = (G', s, t, k, \ell, A, \lambda, K, p)$ denote the obtained instance of VW-SSP. Observe that due to Reduction Rule 4.8, $I'$ is simple (with $A = 0$, see Definition 4.2). Due to Observation 4.9, we know that $G'$ has at most $8|Y| - 7$ vertices and $8|Y| - 8 + fes$ edges, where each vertex is of weight $O(k + \ell)$. Note that $|Y| \leq 2fes + 2$. Hence, $G'$ has at most $16fes + 9$ vertices, $17fes + 8$ edges, and vertex weights in $O(k + \ell)$.

Having shown Proposition 4.6, we can now prove Theorem 4.5. Herein, to strip our shrunk VW-SSP instances of weights, we will employ Proposition 4.3 for Theorem 4.5(i) and Lemma 2.13 for Theorem 4.5(ii).
Proof of Theorem 4.5. Let \( I = (G, s, t, k, \ell) \) be an instance of SSP. Employ Proposition 4.6 to obtain simple instance \( I' = (G', s', t, k, \ell, \lambda, \kappa, \eta) \) of VW-SSP, where \( G' \) has at most \( O(fes) \) vertices and edges, where each vertex is of weight \( O(k + \ell) \). Employing Proposition 4.3 yields an instance \( I'' = (G'', s', t', k', \ell') \) of SSP in time \( M = \kappa(V(G')) + \eta(V(G')) \in O(fes \cdot (k + \ell)). \)

Due to Proposition 4.3, it follows that \( G'' \) has at most \( M \) vertices, yielding (i).

For statement (ii), apply Lemma 2.13 to obtain an instance \( I''' = (G', s', t, k', \ell', \lambda', \kappa', \eta') \) of VW-SSP with \( k', \ell' \), and all weights encoded with \( O(fes^3) \) bits. Since VW-SSP is NP-complete, there is a polynomial-time many-one reduction to SSP. Employing such a polynomial-time many-one reduction on instance \( I''' \), yields statement (ii), since it can blow up the instance size by at most a polynomial. □

4.1.2 A problem kernel with \( O(fvs \cdot (k + \ell) \cdot \max(k, \ell)) \) vertices

We show that SSP admits a problem kernel with a number of vertices quadratic in the parameter \( fvs \).

**Theorem 4.10. Short Secluded Path admits a problem kernel of size polynomial in \( fvs + k + \ell \) with \( O(fvs \cdot (k + \ell) \cdot \max(k, \ell)) \) vertices.**

Let \( G = (V, E) \) be the input graph with \( V = F \cup W \) such that \( F \) is a feedback vertex set with \( s, t \in F \) (hence, \( G[W] \) is a forest). Let \( \beta := |F| \). We distinguish the following types of vertices of \( G \) (see Figure 4.1 for an illustration).

- **\( R \subseteq F \)** is the subset of vertices in \( F \) with more than \( k + \ell \) neighbors or more than \( \ell \) degree-one neighbors. Since no vertex of \( R \) is part of any solution path, we refer to the vertices in \( R \) as **forbidden**.
- **\( Y \subseteq W \)** is the subset of vertices in \( W \) containing all vertices \( v \) with \( N(v) \cap F \not\subseteq R \), that is, vertices that have at least one neighbor in \( F \) that is not forbidden. We call the vertices in \( Y \) **good**.
- **\( T \)** is the set of connected components of \( H := G[W] \), all of which are trees.

Towards proving Theorem 4.10, we will first prove the following, and then strip the weights using Proposition 4.3.

**Proposition 4.11.** For any instance of SSP we can compute in polynomial time an equivalent simple instance of VW-SSP with \( O(fvs \cdot (k + \ell)) \) vertices, \( O(fvs^3 \cdot (k + \ell)) \) edges and vertex weights \( O(k + \ell) \).

We will interpret the input SSP instance as an instance of VW-SSP with vertex weights \( \lambda: V \rightarrow N \cup \{0\}, v \mapsto 1 \), \( \kappa: V \rightarrow N, v \mapsto 1 \), and \( \eta: V \rightarrow N \cup \{0\}, v \mapsto 0 \). For an exemplified illustration of the following reduction rules, refer to Figure 4.1. The first reduction rule ensures that each forbidden vertex remains forbidden throughout our application of all reduction rules. It is clearly applicable in linear time.

**Reduction Rule 4.12.** For each \( v \in R \), set \( \eta(v) = \ell + 1 \).

Since every vertex in \( F \setminus R \) has degree at most \( k + \ell \), by the definition of good vertices, we have the following.

**Observation 4.13.** The number of good vertices is \( |Y| \leq \beta(k + \ell) \).

Since a solution path has neither vertices nor neighbors in any tree \( T \in T \) that does not contain vertices of \( Y \), we delete such trees.

**Reduction Rule 4.14.** Delete all trees \( T \in T \) with \( V(T) \cap Y = \emptyset \).

Note that if Reduction Rule 4.14 is not applicable, then each tree in \( T \) contains a vertex from \( Y \), which gives \( |T| \leq \beta(k + \ell) \) together with Observation 4.13.

The following data reduction rule deletes degree-one vertices in trees that are not in \( Y \), since they cannot be part of a solution path (yet can neighbor it).

**Reduction Rule 4.15.** If there is a tree \( T \in T \) and \( v \in V(T) \setminus Y \) with \( N_T(v) = \{w\} \), then set \( \eta(w) := \min(\ell + 1, \eta(w) + 1) \) and delete \( v \).

Note that updating \( \eta(w) \) to the minimum of \( \ell + 1 \) and \( \eta(w) + 1 \) is correct: if a vertex has any weight at least \( \ell + 1 \), the vertex is equally excluded from any solution path as having weight \( \ell + 1 \).
The next data reduction rule shrinks maximal-edgy paths.
vertices enclosed in the light-gray rectangle are all vertices in the feedback vertex set obtained from Reduction Rule 4.17.

If Reduction Rules 4.14 and 4.15 are not applicable, then every maximal path of degree-two vertices and a path in is a yes-instance if and only if.

Let be an instance of VW-SSP and let be a solution path in . Moreover, we have .

\[ (\Rightarrow) \text{ Let } I \text{ be a yes-instance and } P \text{ be a solution path in } G. \text{ Note that, by construction of } G', \text{ for each } X \subseteq V(G) \setminus (R \cup V(Q)), \text{ we have } N_G(X) = N_G(X). \text{ Thus, if } V(Q) \cap V(P) = 0, \text{ then } P \text{ is also a solution path in } G'. \text{ Hence, assume that } V(Q) \cap V(P) \neq 0. \text{ Since } Q \text{ contains no good vertex and no vertex of degree at least three in } T, \text{ it follows that } V(Q) \subseteq V(P). \text{ Moreover, we have } \kappa'(x) = \kappa(K) \leq k \text{ and } \eta'(x) = \eta(K) \leq \ell. \text{ For the path } P' \text{ in } G' \text{ obtained from } P \text{ by replacing } V(Q) \text{ by } V(Q'), \text{ we have}

\[ \kappa'(V(P')) = \kappa'(V(P') \setminus \{x\}) + \kappa'(x) = \kappa(V(P') \setminus K) + \kappa(K) = \kappa(V(P)), \]
\[ N_G(V(P')) = (N_G(V(P') \setminus \{x\}) \cup (N_G(x) \setminus \{a, b\})) \cup (N_G(K) \setminus \{a, b\}) = N_G(V(P)), \]
\[ \eta'(V(P')) = \eta'(x) + \eta'(V(P') \setminus \{x\}) = \eta(K) + \eta(V(P) \setminus K) = \eta(V(P)). \]  

\[ \text{(\Rightarrow)} \text{ Let } I' \text{ be a yes-instance and } P' \text{ be a solution path in } G'. \text{ If } V(Q') \cap V(P') = 0, \text{ then } P' \text{ is also a solution path in } G. \text{ Hence, assume that } V(Q') \cap V(P') \neq 0. \text{ Since } Q' \text{ contains no good vertex and no vertex of degree at least three in } T, \text{ it follows that } V(Q') \subseteq V(P'). \text{ Moreover, we have } \kappa'(x) = \kappa(K) \leq k \text{ and } \eta'(x) = \eta(K) \leq \ell. \text{ Let } P' \text{ be the path in } G \text{ obtained from } P' \text{ by replacing } V(Q') \text{ by } V(Q). \text{ We have } \kappa(V(P)) = \kappa'(V(P')), \text{ } N_G(V(P)) = N_G(V(P')), \text{ and } \eta(V(P)) = \eta'(V(P')) \text{ by (4.1).} \]

Lemma 4.18. If Reduction Rules 4.14 and 4.15 are not applicable, then Reduction Rule 4.17 is exhaustively applicable in linear time. Moreover, no application of Reduction Rule 4.17 makes Reduction Rule 4.14 or Reduction Rule 4.15 applicable.

Proof. If Reduction Rules 4.14 and 4.15 are not applicable, then every maximal path of degree-two vertices in not containing good vertices is a maximal-edgy path. Hence, employ the following. Let be the set of
all degree-two vertices in \(G - F\) and \(Z'\) be a working copy of \(Z\). As long as \(Z' \neq \emptyset\), do the following. Select any vertex \(v \in Z'\) and start a breadth-first search that stops when a good vertex or a vertex of degree at least three is found. Apply Reduction Rule 4.17 on the just identified maximal-edgy path (if it contains more than three vertices). Delete all the vertices found in the iteration from \(Z'\).

Since no application of Reduction Rule 4.17 deletes a good vertex or creates a vertex of degree one, no application of Reduction Rule 4.17 makes Reduction Rule 4.14 or Reduction Rule 4.15 applicable. \(\Box\)

We prove next that if none of Reduction Rules 4.14, 4.15 and 4.17 is applicable, the trees are small in the sense that

\[
|V(T)| \leq O(3(k + \ell)),
\]

where the last inclusion follows from Observation 4.13. It follows that there are \(|V(T)| \leq O(6(k + \ell)^2\cdot(\ell + 1))\) vertices in each of degree two vertices in \(G\), none of which are good. Since every vertex in \(V(T)\) or \(V(T')\) of degree two is contained in a maximal-edgy path, this contradicts the definition of set \(Q\). It follows that \(|V(T)| \leq O(3(k + \ell)^2\cdot(\ell + 1))\) vertices.

Let \(Y_T := Y \cap V(T)\) denote the set of good vertices in \(T\). Then \(T\) has \(O(3(k + \ell))\) vertices, each of weight \(O(k + \ell)\).

Proof of Proposition 4.11. Compute a feedback vertex set \(F\) of size \(\beta \leq 4fvs\) in linear time \([1]\). Apply all reduction rules exhaustively in linear time: first apply Reduction Rules 4.14 and 4.15 exhaustively in linear time (Lemma 4.16), then Reduction Rule 4.17 exhaustively in linear time (Lemma 4.18).

Now, consider a graph \(G\) to which no data reduction rules are applicable and let \(T_1, \ldots, T_h\) denote the trees in \(G - F\). By Lemma 4.19, each \(T_i\) has \(O(|Y_{T_i}|)\) vertices, each of maximal weight \(O(k + \ell)\), where \(Y_{T_i} = Y \cap V(T_i)\). Thus, the number of vertices and edges in \(G - F\) is

\[
\sum_{i=1}^{h} O(|Y_{T_i}|) = \sum_{i=1}^{h} O(|Y_{T_i}|) = O(|Y|) \subseteq O(\beta \cdot (k + \ell)),
\]

where the last inclusion follows from Observation 4.13. It follows that there are \(O(\beta^2 \cdot (k + \ell))\) edges in \(G\). Altogether, \(G\) has \(O(\beta \cdot (k + \ell))\) vertices, each of weight \(O(k + \ell)\), and \(O(\beta^2 \cdot (k + \ell))\) edges. Moreover, the obtained instance is simple (with \(A = R\), see Definition 4.2). \(\Box\)

Combining Proposition 4.11 with Proposition 4.3, we now prove the main result of this section.

Proof of Theorem 4.10. Let \(I = (G, s, t, k, \ell)\) be an instance of SSP. First, employ Proposition 4.11 to obtain simple instance \(I' = (G', s, t, k, \ell, \lambda, \kappa, \eta)\) of VW-SSP. Then employ Proposition 4.3 to obtain instance \(I'' = (G'', s', t', k', \ell)\) of SSP. We know that \(G''\) has \(O(fvs \cdot (k + \ell))\) vertices, \(O(fvs^2 \cdot (k + \ell))\) edges and vertex weights at most \(O(k + \ell)\). Hence,

\[
|V(G'')| = (\kappa(V(G'')) + \eta(V(G''))) \in O(fvs \cdot (k + \ell) \cdot k + fvs \cdot (k + \ell) \cdot \ell) \subseteq O(fvs \cdot (k + \ell) \cdot \max[k, \ell]).
\] \(\Box\)
4.2 Limits of data reduction

In the previous section, we proved a problem kernel for Short Secluded Path with size polynomial in $fvs + k + \ell$. In this section, we prove that presumably we cannot drop $k$ to obtain a similar result. That is, we prove that SSP when parameterized by $fvs + \ell$, presumably, does not admit a polynomial kernel.

**Theorem 4.20.** Unless coNP $\subseteq$ NP/poly, Short Secluded Path admits no problem kernel with size polynomial in $fvs + \ell$.

To prove Theorem 4.20, we use a cross composition (Definition 3.14) of Multicolored Clique (Problem 2.15) into SSP. In fact, we will reduce from the NP-hard [15, 22] special case of Multicolored Clique where instances $G = (V, E)$ with $V = V_1 \cup \cdots \cup V_k$ and $E_{i,j} := \{(u,v) \in E \mid u \in V_i, v \in V_j\}$ satisfy $|V_i| = |V_j|$ for $1 \leq i < j \leq k$ and $|E_{i,j}| = |E_{i', j'}|$ for all $1 \leq i < j \leq k$ and $1 \leq i' < j' \leq k$. For readability, we often omit the partitions of $V$ and $E$ from the instance description. For the cross composition, we use the following polynomial equivalence relation on instances of Multicolored Clique.

**Lemma 4.21.** Let two Multicolored Clique instances $G = (V, E)$ and $G' = (V', E')$ with $V = V_1 \cup \cdots \cup V_k$ and $V' = V'_1 \cup \cdots \cup V'_k$ be $R$-equivalent if and only if $|V| = |V'|$, $|E| = |E'|$, and $k = k'$. Then, $R$ is a polynomial equivalence relation.

**Proof.** Deciding whether two instances $G = (V, E)$ and $G' = (V', E')$ are $R$-equivalent is doable in $O(|V| + |V'| + |E| + |E'|)$ time. Now, let $S \subseteq \Sigma^*$ be a set of instances and $G = (V, E)$ with $n = |V|$ and $m = |E|$ be such that $n + m = \max_{x \in S} |x|$. There are at most $n + m$ different vertex set sizes, edge set sizes, and partition sizes of the vertex sets, resulting in at most $(n + m)^\ell$ equivalence classes. □

We next describe the cross composition.

**Construction 4.22.** Let $G_1 = (V_1, E_1), \ldots, G_p = (V_p, E_p)$ be $p = 2^q$ Multicolored Clique instances equivalent under $\mathcal{R}$, where $q \in \mathbb{N}$. Then, we can denote $n := |V_1|$, $m := |E_1|$, $V_a = V_{a,1} \cup \cdots \cup V_{a,k}$ with $V_{a,j} := \{v_{a,j}^1, \ldots, v_{a,j}^{k_j}\}$, and $E_a = \bigcup_{1 \leq j \leq k} E_{a,j}$ with $E_{a,j} = \{e_{a,j}^1, \ldots, e_{a,j}^{k_j}\}$ for all $a \in \{1, \ldots, p\}$. Construct the following SSP instance $(G, s, t, \ell, \ell')$ with graph $G$ (refer to Figure 4.2 for an illustration). Let $G$ be initially empty, $L := p \cdot n \cdot m$, and $K := \binom{k}{2}$.

1. Add $q + 1$ paths $A_1, \ldots, A_q+1$, where $V(A_i) = \{a_{i,1}, \ldots, a_{i,q}\}$ and $a_{i,1}, a_{i,q}$ are the end points. For each $y \in \{1, \ldots, q\}$, add two vertices $u_{i,y}$ and $u_{i,y+1}$, and make each of them adjacent to $a_{i,y}$ and $a_{i,y+1}$. Define $U = \{u_{i,y} \mid y \in \{1, \ldots, q\}, i \in \{1, 2\}\}$. Refer to Figure 4.2b).

2. Add $K + 1$ paths $B_1, \ldots, B_K+1$, where $V(B_i) = \{b_{i,1}, \ldots, b_{i,K}\}$ and $b_{i,1}, b_{i,K}$ are the end points. For each $y \in \{1, \ldots, K\}$, add the vertex set $F_y := \{e_1^y, \ldots, e_r^y\}$ and make each vertex in $F_y$ adjacent to $b_{y,1}$ and $b_{y,K}$. Define $F = \bigcup_{y=1}^{K} F_y$. Choose an arbitrary bijection $\pi : \{1, \ldots, K\} \rightarrow \{i, j \mid 1 \leq i < j \leq K\}$. We say that $e_i^y$ corresponds to the $z$-th edge $e'_{\pi(z)} \in E_{a,\pi(z)}$ for all $a \in \{1, \ldots, p\}$. Refer to Figure 4.2c).

3. Add $p$ paths $P_1, \ldots, P_p$ such that $P_a$ has vertex set $\{v_{a,i}^k \mid i \in \{1, \ldots, k\}, y \in \{1, \ldots, r\}\}$ and edge set $\{(v_{a,i}^k, v_{a,i+1}^k) \mid i \in \{1, \ldots, k - 1\}\}$ or $\{(v_{a,i}^k, v_{a,i}^{k+1}) \mid y \in \{1, \ldots, r - 1\}\}$. We say $P_a$ corresponds to the vertices in $V_a$ in the $a$-th graph $G_a$. Next, for each $a \in \{1, \ldots, p + 1\}$, add a path of three vertices $w_{a,1}, w_{a,2}, w_{a,3}$ with edges $(w_{a,1}, w_{a,2}), (w_{a,2}, w_{a,3})$. Make $w_{a,1}$ adjacent to $a_{q+1,1}$ and $w_{p+1,3}$ adjacent to $b_{1,1}$. For each $1 \leq a \leq p + 1$, make $w_{a,1}$ adjacent to $v_{a,1}^k$. For each $1 \leq a < p + 1$, make $w_{a,3}$ adjacent to $v_{a,1}^k$.

4. Add one vertex $h$ and make each $w_{a,2}$ adjacent to $h$.

5. For each $a \in \{1, \ldots, p\}$, make each $v \in V(P_a)$ adjacent to the vertex corresponding to an incident edge in $F$. That is, if $v$ is incident with edge $e'_{\pi(z)}$, make $v$ adjacent to vertex $e_i^y$.

6. For each $a \in \{1, \ldots, p\}$, make each $v \in V(P_a)$ adjacent to the vertices in $U$ as follows: we associate $u_{a,1}$ with a 0-1-vector of length $p$ with alternating blocks of size $p/2^q$, starting with a block of ones, and $u_{a,2}$ is the complement 0-1-vector of $u_{a,1}$. Vertex $v \in V(P_a)$ is adjacent to $u$ if $a$ is at $a$-th position in the 0-1-vector associated with $u$.

7. Add $s$ and $t$. Make $t$ adjacent to $b_{K+1,1}$. Make $s$ adjacent to all vertices except the vertices in $\bigcup_{a=1}^p V(P_a)$.
Latter, we observe that the first vertex on the inner vertices of Lemma 4.23.

Before we prove that the instance is a yes-instance, we prove some crucial properties of solutions in I in the case that I is a yes-instance.

Lemma 4.23. Let \((G, s, t, k’, \ell)\) be the SSP-instance obtained from Construction 4.22 and let \((G, s, t, k’, \ell)\) be a yes-instance. Let \(P\) be a solution \(s-t\) path in \(G\). Then the following hold:

(i) \(P\) contains each path \(Q \in \{A_2, \ldots, A_{q+1}, B_1, \ldots, B_{K+1}\}\) and a subpath of \(A_1\) as subpath. Moreover, the first vertex on \(P\) after \(s\) is in \(V(A_1) \setminus \{a_{1,L}\}\).

(ii) \(|V(P) \cap U_z| = |V(P) \cap F_z| = 1\) for all \(y \in \{1, \ldots, q\}\), \(z \in \{1, \ldots, K\}\).

(iii) Let \(v\) be a vertex of some \(U_y\) \((F_z)\) contained in \(P\), and let \((\{v’, v, v’‘\}, \{v, v’, v’‘\})\) be the length-3 subpath in \(P\). Then \(v’ = a_{j,L}\) and \(v’‘ = a_{j+1,1}\) \((v’ = b_{j,L}\) and \(v’‘ = b_{j+1,1}\))

Proof. (i): From each path \(Q \in \{A_2, \ldots, A_{q+1}, B_1, \ldots, B_{K+1}\}\), at least \(L - \ell > \ell\) vertices must be contained. Since the inner vertices of \(Q\) are only adjacent to vertices in \(Q\) and \(s\), it follows that \(Q\) is a subpath of \(P\). Moreover, also at least \(L - \ell > \ell\) vertices from \(A_1\) must be contained in \(P\). Hence, a subpath of \(A_1\) is a subpath of \(P\). From the latter, we observe that the first vertex on \(P\) after \(s\) is in \(V(A_1) \setminus \{a_{1,L}\}\).

(ii): From (i), we know that each path \(Q \in \{A_2, \ldots, A_{q+1}, B_1, \ldots, B_{K+1}\}\) is a subpath of \(P\), and the first vertex on \(P\) after \(s\) is in \(V(A_1) \setminus \{a_{1,L}\}\). If \(Q = A_y\), \(2 \leq y \leq q + 1\), we know that \(a_{y,1}\) is only incident with vertices in \(U_{y-1} \cup \{s\} \cup \{a_{y,2}\}\). It follows that for each \(U_y\) at least one vertex is contained in \(P\). If \(Q = B_z\), \(2 \leq z \leq K + 1\),
we know that \(b_{i,j}\) is only incident with vertices in \(F_{i-1} \cup \{s\} \cup \{b_{i,2}\}\). It follows that for each \(F_i\), at least one vertex is contained in \(P\). Suppose there is a set \(X \subseteq \{U_1, \ldots, U_q, F_1, \ldots, F_K\}\) such that at least two vertices from \(X\) are contained in \(P\). Recall that by construction, each vertex in \(U \cup F\) has \(M^2\) degree-one neighbors. Then \(P\) has at least \(M^2 |A_1| + M^2 \log(p) + M^2 > t\) neighbors, yielding a contradiction. Hence, we know that for each \(U_i\) and \(F_j\), exactly one vertex is contained in \(P\).

(iii): Let \(v \in U_i\). Suppose that \(v' \not \in A_i\) (for \(v'\), this works analogously). We know that \(P\) contains \(A_i\) as a subpath. Hence, \(A_i\) is adjacent to the other vertex in \(U_i \setminus \{v\}\) on \(P\), yielding a contradiction to (ii).

In the same way, we can prove the claim for \(v \in F_j\). \(\square\)

We proceed proving that the instance obtained from Construction 4.22 is a yes-instance if and only if at least one input instance is a yes-instance.

**Lemma 4.24.** Let \((G_a)_{a=1, \ldots, p}\) be \(p = 2^q\) instances of Multicolored Clique that are \(R\)-equivalent, where \(q \in \mathbb{N}\). Let \((G, s, t, k', \ell)\) be the SSP-instance obtained from Construction 4.22. Then at least one instance \(G_a\) is a yes-instance if and only if \((G, s, t, k', \ell)\) is yes-instance for SSP.

**Proof.** (\(\Rightarrow\)) Let \(G_a\) be a yes-instance for some \(a \in \{1, \ldots, p\}\) and let \(C\) be a \(k\)-clique in \(G_a\). Construct an \(s\)-\(t\) path \(P\) as follows: \(P\) starts at \(s\), then goes to \(a_1\), follows along the vertices only in \(A_1, \ldots, A_{p+1}\) and \(U\) until \(a_{p+2}\), while selecting the vertices in \(U\) such that only the vertices corresponding to \(V(G_a)\) are not in the neighborhood yet. This is possible since, for each \(b \in \{1, \ldots, p\}\), only one of \(u_1, u_2\) is adjacent to the vertices in \(V(P_b)\). Next, follow the vertices in \(V(P_1), \ldots, V(P_p)\), avoiding the vertices in \(V(P_a)\) by using \(w_{a_2}, w_{a_{1,2}}\). Then follow, starting at \(b_{1,1}\) towards \(b_{p+1,1}\) and then to \(i\) by only selecting the vertices corresponding to the edges in \(C\). This path contains \(1 + (q+1)\ell + \log(p) + (p-1)n + (3p+1) - 1 + (K+1) + (K+1) \leq k'\) vertices. Moreover, the path has \(M + M^2 (\ell) + M^2 \log(p) \leq \ell\) neighboring vertices.

(\(\Leftarrow\)) Let \((G, s, t, k', \ell)\) be a yes-instance for SSP. Let \(P\) be a solution \(s\)-\(t\) path. We claim that if \(P\) contains a vertex in \(V(P_a)\) for some \(a \in \{1, \ldots, p\}\), then it contains all vertices in \(V(P_a)\). Suppose not, that is, there is an \(a \in \{1, \ldots, p\}\) such that \(1 \leq |V(P) \cap V(P_a)| < n\). Note that \(|N(V(P_a))| \subseteq U \cup F \cup \{w_{a_2}, w_{a_{1,2}}\}\). Since \(1 \leq |V(P) \cap V(P_a)| < n\), there is a vertex \(v \in V(P_a) \cap V(P)\) such that at least one of its neighbors in \(V(P)\) is not contained in \(V(P)\). It follows that \(v\) is adjacent to a vertex in \(U \cup F\). This contradicts Lemma 4.23(iii).

By the values of \(k'\) and \(\ell\), we know that either exactly one \(P_a\) is not contained in \(P\), or there are \(n + 2\) vertices from \(A_1\) being not contained in \(P\). In the latter case, we have at least \(n + |E| - (\ell) + \log(p) + M^2 (\ell) + M^2 \log p \leq M + M^2 (\ell) + M^2 \log p = \ell\) neighbors (we can assume that \(n > k + 2\)), yielding a contradiction. It follows the former case: there is exactly one \(P_a\) not contained in \(P\). It follows that \(h \in V(P)\).

By Lemma 4.23(ii), from each \(F_i\) there is exactly one vertex contained in \(P\). Moreover, for each \(y \in \{1, \ldots, K\}\) and for each \(v \in F_i\), it holds true that \(|V(P_a) \cap N(v)| \geq 2\). Hence, \(|N(V(P)) \cap V(P_a)| \geq k\), as \(K\) edges cannot be distributed among less than \(k\) vertices. It follows that \(A_1\) is a subpath of \(P\).

Since \(|N(V(P)) \cap V(P_a)| = |E| - (\ell) + \log(p) + 2 + M^2 (\ell) + M^2 \log(p)\), it follows that there must be exactly \(k\) vertices in \(V(P_a)\) neighboring \(P\). This witnesses a \(k\)-clique in \(G_a\), and the statement follows. \(\square\)

We are ready to prove the main result of this section.

**Proof of Theorem 4.20.** Due to Lemma 4.21, we know that \(R\) is a polynomial equivalence relation on the instances of Multicolored Clique. Let \(G_1 = (V_1, E_1), \ldots, G_p = (V_p, E_p)\) be \(p = 2^q\) instances of Multicolored Clique that are \(R\)-equivalent, where \(q \in \mathbb{N}\). We construct an instance \((G, s, t, k', \ell)\) of SSP by applying Construction 4.22 in time polynomial in \(\sum_{a=1}^{p} (|V_a| + |E_a|)\). By Lemma 4.24, we have that \((G, s, t, k', \ell)\) is a yes-instance if and only if \((G_a, k)\) is a yes-instance for some \(a \in \{1, \ldots, p\}\). The set \(W := U \cup F \cup \{s, h, t\}\) forms a feedback vertex set with \(|W| \leq 2 \log(p) + K \cdot x\), that is, \(|W|\) is upper-bounded by a polynomial in \(|V_a| + |E_a| + \log(p)\) for any \(a \in \{1, \ldots, p\}\). Moreover, \(\ell = M + M^2 (\ell) + M^2 \log(p)\), where \(M := k + |E| - (\ell) + \log(p) + 2\) is upper-bounded by a polynomial in \(|V_a| + |E_a| + \log(p)\). Altogether, we described a cross composition from Multicolored Clique into SSP parameterized by \(fvs + \ell\), and the statement follows. \(\square\)

**5 Conclusion**

Concluding, we point out that our algorithms for VW-SSP on graphs of bounded treewidth (Theorem 3.2) can easily be generalized to a problem variant where also edges have a weight counting towards the path length, and so
can our subexponential-time algorithms in planar graphs (Theorem 2.1). Moreover, the technique of Bodlaender et al. [10] that our algorithm is based on has experimentally been proven to be practically implementable [21].

In contrast, we observed SSP to be a problem for which provably effective polynomial-time data reduction is rather hard to obtain (Theorems 2.14, 3.13 and 4.20). Therefore, studying relaxed models of data reduction with performance guarantees like approximate [23, 40] or randomized kernelization [39] seems worthwhile.

Indeed, some of our positive results on kernelization, in particular our problem kernels of size \( c^{O(r)} \) in \( K_{r,r} \)-free graphs and of size \( fes^{O(1)} \) in graphs of feedback edge number \( fes \) for SSP (Theorems 2.5 and 4.5), for now, can be mainly seen as a proof of concept, since they employ the quite expensive weight reduction algorithm of Frank and Tardos [26] and we have no “direct” way of reducing VW-SSP back to SSP. On the positive side, our solution algorithms also work for VW-SSP, so that they can be applied to the linear-time computable weighted shrunk instances and stripping the weights is not necessary from a practical point of view.

Acknowledgments. This research was initiated during a research visit of René van Bevern and Oxana Yu. Tsidulko at TU Berlin. Till Fluschnik was supported by the German Research Foundation, grant NI 369/18. René van Bevern and Oxana Yu. Tsidulko were supported by the Russian Foundation for Basic Research (RFBR), grant 18-501-12031 NNIO_a, while working on Sections 2.2, 2.3, 3.2 and 4. While working on Sections 2.1 and 3.1, René van Bevern was supported by RFBR grant 16-31-60007 mol_a_dk and Oxana Yu. Tsidulko was supported by the Ministry of Science and Higher Education of the Russian Federation under the 5-100 Excellence Programme.

References

[1] Reuven Bar-Yehuda, Dan Geiger, Joseph Naor, and Ron M. Roth. Approximation algorithms for the feedback vertex set problem with applications to constraint satisfaction and bayesian inference. SIAM J. Comput., 27 (4):942–959, 1998. doi:10.1137/S0097539796305109.

[2] Hannah Bast, Daniel Delling, Andrew Goldberg, Matthias Müller-Hannemann, Thomas Pajor, Peter Sanders, Dorothea Wagner, and Renato F. Werneck. Route planning in transportation networks. In Algorithm Engineering: Selected Results and Surveys, volume 9220 of Lecture Notes in Computer Science, pages 19–80. Springer, 2016. doi:10.1007/978-3-319-49487-6_2.

[3] René van Bevern, Till Fluschnik, and Oxana Yu. Tsidulko. Parameterized algorithms and data reduction for safe convoy routing. In Proceedings of the 18th Workshop on Algorithmic Approaches for Transportation Modeling, Optimization, and Systems, volume 65 of OpenAccess Series in Informatics (OASIcs), pages 10:1–10:19. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2018. doi:10.4230/OASIcs.ATMOS.2018.10.

[4] René van Bevern, Rolf Niedermeier, Manuel Sorge, and Mathias Weller. Complexity of arc routing problems. In Arc Routing: Problems, Methods, and Applications, volume 20 of MOS-SIAM Series on Optimization. SIAM, 2014. doi:10.1137/1.9781611973679.ch2.

[5] René van Bevern, Christian Komusiewicz, and Manuel Sorge. A parameterized approximation algorithm for the mixed and windy capacitated arc routing problem: Theory and experiments. Networks, 70(3):262–278, 2017. doi:10.1002/net.21742.

[6] René van Bevern, Till Fluschnik, George B. Mertzios, Hendrik Molter, Manuel Sorge, and Ondřej Suchý. The parameterized complexity of finding secluded solutions to some classical optimization problems on graphs. Discrete Optimization, 2018. doi:10.1016/j.disopt.2018.05.002. In press, available on arXiv:1606.09000v5.

[7] Hans L. Bodlaender. A linear-time algorithm for finding tree-decompositions of small treewidth. SIAM Journal on Computing, 25(6):1305–1317, 1996. doi:10.1137/S0097539793251219.

[8] Hans L. Bodlaender, Rodney G. Downey, Michael R. Fellows, and Danny Hermelin. On problems without polynomial kernels. Journal of Computer and System Sciences, 75(8):423–434, 2009. doi:10.1016/j.jcss.2009.04.001.

[9] Hans L. Bodlaender, Bart M. P. Jansen, and Stefan Kratsch. Kernelization lower bounds by cross-composition. SIAM Journal on Discrete Mathematics, 28(1):277–305, 2014. doi:10.1137/120880240.
[10] Hans L. Bodlaender, Marek Cygan, Stefan Kratsch, and Jesper Nederlof. Deterministic single exponential time algorithms for connectivity problems parameterized by treewidth. *Information and Computation*, 243:86–111, 2015. doi:10.1016/j.ic.2014.12.008.

[11] Hans L. Bodlaender, Pál Grönnás Drange, Markus S. Dregi, Fedor V. Fomin, Daniel Lokshstanov, and Michał Pilipczuk. A $2^kn$-approximation algorithm for treewidth. *SIAM Journal on Computing*, 45(2):317–378, 2016. doi:10.1137/130947374.

[12] Drago Bokal, Gasper Fijavz, and Bojan Mohar. The minor crossing number. *SIAM Journal on Discrete Mathematics*, 20(2):344–356, 2006. doi:10.1137/05062706X.

[13] Hans-Joachim Böckenhauer, Juraj Hromkovič, Joachim Kneis, and Joachim Kupke. The parameterized approximability of TSP with deadlines. *Theory of Computing Systems*, 41(3):431–444, 2007. doi:10.1007/s00224-007-1347-x.

[14] Shiri Chechik, Matthew P. Johnson, Merav Parter, and David Peleg. secluded connectivity problems. *Algorithmica*, 79(3):708–741, 2017. doi:10.1007/s00453-016-0222-z.

[15] Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015. doi:10.1007/978-3-319-21275-3.

[16] Erik D. Demaine and Mohammadtaghi Hajiaghayi. Linearity of grid minors in treewidth with applications through bidimensionality. *Combinatorica*, 28(1):19–36, 2008. doi:10.1007/s00493-008-2140-4.

[17] Reinhard Diestel. *Graph Theory*, volume 173 of *Graduate Texts in Mathematics*. Springer, 4th edition, 2010. doi:10.1007/978-3-662-53622-3.

[18] Frederic Dorn, Hannes Moser, Rolf Niedermeier, and Mathias Weller. Efficient algorithms for Eulerian Extension and Rural Postman. *SIAM Journal on Discrete Mathematics*, 27(1):75–94, 2013. doi:10.1137/110834810.

[19] Rod G. Downey and Michael R. Fellows. *Fundamentals of Parameterized Complexity*. Springer, 2013. doi:10.1007/978-1-4471-5559-1.

[20] Michael Etscheid, Stefan Kratsch, Matthias Mnich, and Heiko Röglin. Polynomial kernels for weighted problems. *Journal of Computer and System Sciences*, 84(Supplement C):1–10, 2017. doi:10.1016/j.jcss.2016.06.004.

[21] Stefan Fafianie, Hans L. Bodlaender, and Jesper Nederlof. Speeding up dynamic programming with representative sets: An experimental evaluation of algorithms for steiner tree on tree decompositions. *Algorithmica*, 71(3):636–660, 2015. doi:10.1007/s00453-014-9934-0.

[22] Michael R. Fellows, Danny Hermelin, Frances A. Rosamond, and Stéphane Vialette. On the parameterized complexity of multiple-interval graph problems. *Theoretical Computer Science*, 410(1):53–61, 2009. doi:10.1016/j.tcs.2008.09.065.

[23] Michael R. Fellows, Ariel Kulik, Frances A. Rosamond, and Hadas Shachnai. Parameterized approximation via fidelity preserving transformations. *Journal of Computer and System Sciences*, 93:30–40, 2018. doi:10.1016/j.jcss.2017.11.001.

[24] Jörg Flum and Martin Grohe. *Parameterized Complexity Theory*. Springer, 2006. doi:10.1007/3-540-29953-X.

[25] Fedor V. Fomin, Petr A. Golovach, Nikolay Karpov, and Alexander S. Kulikov. Parameterized complexity of secluded connectivity problems. *Theory of Computing Systems*, 61(3):795–819, 2017. doi:10.1007/s00224-016-9717-x.

[26] András Frank and Éva Tardos. An application of simultaneous diophantine approximation in combinatorial optimization. *Combinatorica*, 7(1):49–65, 1987. doi:10.1007/BF02579200.

[27] Achille Giacometti. River networks. In *Complex Networks*, Encyclopedia of Life Support Systems (EOLSS), pages 155–180. EOLSS Publishers/UNESCO, 2010.
[28] Petr A. Golovach, Pinar Heggernes, Paloma T. Lima, and Pedro Montealegre. Finding connected secluded subgraphs. In *Proceedings of the 12th International Symposium on Parameterized and Exact Computation, IPEC 2017, September 6-8, 2017, Vienna, Austria*, volume 89 of LIPIcs, pages 18:1–18:13. Schloss Dagstuhl – Leibniz-Zentrum fuer Informatik, 2017. doi:10.4230/LIPIcs.IPEC.2017.18.

[29] Jiong Guo and Rolf Niedermeier. Invitation to data reduction and problem kernelization. *ACM SIGACT News*, 38(1):31–45, 2007. doi:10.1145/1233481.1233493.

[30] G. Gutin and V. Patel. Parameterized traveling salesman problem: Beating the average. *SIAM Journal on Discrete Mathematics*, 30(1):220–238, 2016. doi:10.1137/140980946.

[31] Gregory Gutin, Gabriele Muciaccia, and Anders Yeo. Parameterized complexity of k-Chinese Postman Problem. *Theoretical Computer Science*, 513:124–128, 2013. doi:10.1016/j.tcs.2013.10.012.

[32] Gregory Gutin, Mark Jones, and Magnus Wahlström. The mixed chinese postman problem parameterized by pathwidth and treedepth. *SIAM Journal on Discrete Mathematics*, 30(4):2177–2205, 2016. doi:10.1137/15M1034337.

[33] Gregory Gutin, Mark Jones, and Bin Sheng. Parameterized complexity of the k-arc chinese postman problem. *Journal of Computer and System Sciences*, 84:107–119, 2017. doi:10.1016/j.jcss.2016.07.006.

[34] Gregory Gutin, Magnus Wahlström, and Anders Yeo. Rural Postman parameterized by the number of components of required edges. *Journal of Computer and System Sciences*, 83(1):121–131, 2017. doi:10.1016/j.jcss.2016.06.001.

[35] Danny Hermelin, Stefan Kratsch, Karolina Soltys, Magnus Wahlström, and Xi Wu. A completeness theory for polynomial (turing) kernelization. *Algorithmica*, 71(3):702–730, 2015. doi:10.1007/s00453-014-9910-8.

[36] Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity? *Journal of Computer and System Sciences*, 63(4):512–530, 2001. doi:10.1006/jcss.2001.1774.

[37] Philip N. Klein and Daniel Marx. A subexponential parameterized algorithm for Subset TSP on planar graphs. In *Proceedings of the 25th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA’14)*, pages 1812–1830. Society for Industrial and Applied Mathematics, 2014. doi:10.1137/1.9781611973402.131.

[38] Stefan Kratsch. Recent developments in kernelization: A survey. *Bulletin of the EATCS*, 113, 2014. [http://eatcs.org/beatcs/index.php/beatcs/article/view/285](http://eatcs.org/beatcs/index.php/beatcs/article/view/285).

[39] Stefan Kratsch and Magnus Wahlström. Compression via matroids: A randomized polynomial kernel for odd cycle transversal. *ACM Transactions on Algorithms*, 10(4):20:1–20:15, 2014. doi:10.1145/2635810.

[40] Daniel Lokshhtanov, Fahad Panolan, M. S. Ramanujan, and Saket Saurabh. Lossy kernelization. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing (STOC 2017)*, pages 224–237. ACM, 2017. doi:10.1145/3055399.3055456.

[41] Max-Jonathan Luckow and Till Fluschnik. On the computational complexity of length- and neighborhood-constrained path problems. Available on arXiv:1808.02359, 2018.

[42] Ross M. McConnell and Jeremy P. Spinrad. Modular decomposition and transitive orientation. *Discrete Mathematics*, 201(1):189–241, 1999. doi:10.1016/S0012-365X(98)00319-7.

[43] Neeldhara Misra, Venkatesh Raman, and Saket Saurabh. Lower bounds on kernelization. *Discrete Optimization*, 8(1):110–128, 2011. doi:10.1016/j.disopt.2010.10.001.

[44] Rolf Niedermeier. *Invitation to Fixed-Parameter Algorithms*. Oxford University Press, 2006. doi:10.1093/acprof:oso/9780198566076.001.0001.

[45] Janos Pach, Rados Radoicic, Gabor Tardos, and Geza Toth. Improving the crossing lemma by finding more crossings in sparse graphs. *Discrete & Computational Geometry*, 36(4):527–552, 2006. doi:10.1007/s00454-006-1264-9.
[46] Manuel Sorge, René van Bevern, Rolf Niedermeier, and Mathias Weller. From few components to an Eulerian graph by adding arcs. In Proceedings of the 37th International Workshop on Graph-Theoretic Concepts in Computer Science (WG’11), volume 6986 of Lecture Notes in Computer Science, pages 307–318. Springer, 2011. doi:10.1007/978-3-642-25870-1_28.

[47] Manuel Sorge, René van Bevern, Rolf Niedermeier, and Mathias Weller. A new view on Rural Postman based on Eulerian Extension and Matching. Journal of Discrete Algorithms, 16:12–33, 2012. doi:10.1016/j.jda.2012.04.007.

[48] K. Wagner. Über eine Eigenschaft der ebenen Komplexe. Mathematische Annalen, 114(1):570–590, 1937. doi:10.1007/BF01594196.