On the Parameterized Complexity of Multiway Near-Separator

Bart M. P. Jansen
Eindhoven University of Technology, The Netherlands

Shivesh K. Roy
Eindhoven University of Technology, The Netherlands

Abstract

We study a new graph separation problem called Multiway Near-Separator. Given an undirected graph G, integer k, and terminal set $T \subseteq V(G)$, it asks whether there is a vertex set $S \subseteq V(G) \setminus T$ of size at most k such that in graph $G - S$, no pair of distinct terminals can be connected by two pairwise internally vertex-disjoint paths. Hence each terminal pair can be separated in $G - S$ by removing at most one vertex. The problem is therefore a generalization of (Node) Multiway Cut, which asks for a vertex set for which each terminal is in a different component of $G - S$. We develop a fixed-parameter tractable algorithm for Multiway Near-Separator running in time $2^{\mathcal{O}(k \log k)} \cdot n^{\mathcal{O}(1)}$. Our algorithm is based on a new pushing lemma for solutions with respect to important separators, along with two problem-specific ingredients. The first is a polynomial-time subroutine to reduce the number of terminals in the instance to a polynomial in the solution size k plus the size of a given suboptimal solution. The second is a polynomial-time algorithm that, given a graph G and terminal set $T \subseteq V(G)$ along with a single vertex $x \in V(G)$ that forms a multiway near-separator, computes a 14-approximation for the problem of finding a multiway near-separator not containing x.

2012 ACM Subject Classification Mathematics of computing → Graph algorithms; Theory of computation → Parameterized complexity and exact algorithms; Theory of computation → Approximation algorithms analysis

Keywords and phrases fixed-parameter tractability, multiway cut, near-separator

Digital Object Identifier 10.4230/LIPIcs.IPEC.2023.28

Related Version *Full Version*: `https://arxiv.org/abs/2310.04332`

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

1 Introduction

Graph separation problems play an important role in the study of graph algorithms. While the problem of finding a minimum vertex set whose removal separates two terminals s and t can be solved in polynomial-time via the Ford-Fulkerson algorithm [13], many variations of the problem are NP-complete. They form a fruitful subject of investigation in the study of parameterized algorithmics, where a typical goal is to develop an algorithm that finds a suitable separator of size k in an n-vertex input graph in time $f(k) \cdot n^{\mathcal{O}(1)}$, or concludes that no such solution exists. Landmark results in this area include the FPT algorithms for Multiway Cut [5, 8, 15, 20, 27] (in which the goal is to find a vertex set which separates any pair of terminals from a given set T) and Multicut [3, 22] (in which only a specified subset of the terminal pairs must be separated) in undirected graphs.

18th International Symposium on Parameterized and Exact Computation (IPEC 2023).
Editors: Neeldhara Misra and Magnus Wahlström; Article No. 28; pp. 28:1–28:18
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

After the parameterized complexity of the most fundamental separation problems in this area were settled, researchers started considering variations on the theme of graph separation, including Steiner Multicut [4] (given a sequence of subsets of terminals $T_1, \ldots, T_\ell$, find a vertex set separating at least one pair $\{t_i \in T_i, t_j \in T_j\}$ for each $i \neq j$), Multiway Cut-Uncut [6, 19] (given an equivalence relation $\mathcal{R}$ on a set of terminal vertices, find a vertex set whose removal leaves terminals t_i, t_j in the same connected component if and only if $t_i \equiv_{\mathcal{R}} t_j$), and Stable Multiway cut [21] (find a multiway cut that is independent).

In this paper we start to explore a new variation of the separation theme from the parameterized perspective. Rather than asking for a vertex set which fully separates some given terminal pairs, we are interested in *nearly* separating terminals: in the remaining graph, there should not be *two or more* internally vertex-disjoint paths connecting a terminal pair. We therefore study the following problem, which we believe is a natural extension in the well-studied area of graph separation problems.

Multiway Near-Separator (mwns)	**Parameter:** k
Input: An undirected graph G, terminal set $T \subseteq V(G)$, and a positive integer k.	
Question: Is there a set $S \subseteq V(G) \setminus T$ with $\lvert S\rvert \leq k$ such that there does not exist a pair of distinct terminals $t_i, t_j \in T$ with *two* internally vertex-disjoint t_i-t_j paths in $G - S$?	

Note that, by Menger's theorem, the requirement on solutions S to mwns is equivalent to the requirement that in the graph $G - S$, any terminal pair can be separated by removing a non-terminal vertex. One could therefore imagine applications of this problem in the study of disrupting communications between nodes in a network. While the standard Multiway Cut problem captures the setting that all potential for communication between terminals has to be broken, the size of a solution to the near-separation problem can be arbitrarily much smaller while it still ensures the following property: the communication between each terminal pair is either broken by the solution, or there is at least one non-terminal vertex through which all communications of the pair must pass, so that it may be intercepted at that point. A related problem of reducing connectivity between nodes by removing edges or vertices was studied by Barman and Chawla [1], who presented various approximation algorithms.

Using the (perhaps non-standard) view that a direct edge between two vertices t_i, t_j means there are two internally vertex-disjoint paths between them (the intersection of the set of interior vertices is empty), the requirement that in $G - S$ there do not exist two internally vertex-disjoint paths between any pair of distinct terminals, can alternatively be shown to be equivalent to demanding that T is an independent set and there is no simple cycle containing at least two vertices from T (see Proposition 2.7). Simple cycles containing two vertices from T, henceforth called T-cycles, therefore play an important role in our arguments. Based on this alternative characterization, the near-separator problem is related to the Subset Feedback Vertex Set problem, which asks for a minimum vertex set intersecting all cycles that contain at least one terminal [9], although there the solution is allowed to contain terminal vertices.

A simple reduction (Lemma B.1) shows that Multiway Near-Separator is NP-complete. It forms a generalization of the (Node) Multiway Cut problem with undeletable terminals: an instance $(G, T = \{t_1, \ldots, t_\ell\}, k)$ of the latter problem can be reduced to an equivalent instance (G', T, k) of mwns by inserting $|T|-1$ new non-terminal vertices $w_1, \ldots, w_{|T|-1}$ with $N(w_i) = \{t_i, t_{i+1}\}$.

The Multiway Near-Separator problem can be shown to be non-uniformly fixed-parameter tractable parameterized by k using the technique of recursive understanding [19], as the problem can be formulated in Monadic Second-Order Logic and can be shown to become

fixed-parameter tractable on $(s(k), k+1)$-unbreakable graphs for some function $s\colon \mathbb{N} \to \mathbb{N}$ by branching on small connected vertex sets with small neighborhoods. This only serves as a complexity classification, however, as the resulting algorithms are non-uniform and have a large (and unknown) parameter dependence $f(k)$. In this paper, our goal is to understand the structure of Multiway Near-Separator and develop an efficient (and uniform) parameterized FPT algorithm for the problem.

Our results

The main result of our paper is the following theorem, showing that MWNS has a uniform FPT algorithm with parameter dependence $2^{\mathcal{O}(k \log k)}$.

▶ **Theorem 1.1 (★).** *Multiway Near-Separator can be solved in time* $2^{\mathcal{O}(k \log k)} \cdot n^{\mathcal{O}(1)}$.

The starting point for the algorithm is the approach for solving Multiway Cut via important separators. The *pushing lemma* due to Marx [20] [7, Lemma 8.53] states that for any instance (G, T, k) of Multiway Cut and any choice of terminal $t \in T$, there is an optimal solution which contains an *important* $(t, T \setminus \{t\})$-separator. As the number of important separators of size at most k is bounded by 4^k, we can construct a solution by picking an arbitrary terminal which is not yet fully separated from the remaining terminals and branching on all choices of including a $(t, T \setminus \{t\})$-separator in the solution.

Adapting this strategy directly for Multiway Near-Separator fails for several reasons. Most importantly, since terminal pairs are allowed to remain in the same connected component, it is possible that in an instance with a solution S of size k, there exists a terminal $t \in T$ for which there are no $(t, T \setminus \{t\})$-separators of size $f(k)$ for any function f. This happens when such a terminal t is located in a "central" block in the block-cut tree of $G - S$. However, there always exists a terminal $t' \in T$ for which there does exist a $(t', T \setminus \{t'\})$-separator of size at most $k+1$, and for which furthermore a variation of the pushing lemma can be proven: there is an optimal solution which contains all-but-one vertex of an important $(t', T \setminus \{t'\})$-separator of size at most $k+1$. Intuitively, such a terminal t' can be found in a leaf block of the block-cut tree of $G - S$. Hence there *exists* a terminal for which branching on important separators can make progress in identifying the solution, but not all terminals have this property and a priori it is not clear which is the right one.

To resolve this issue, we will effectively have the algorithm try all choices for the terminal t which is near-separated from all other terminals by an important separator. To ensure the branching factor of the resulting algorithm is bounded in terms of the parameter k, while the number of terminals T may initially be arbitrarily large compared to k, we therefore have to reduce the number of terminals to $k^{\mathcal{O}(1)}$ in a preprocessing phase.

For the standard Multiway Cut problem, a preprocessing step based on the linear-programming relaxation of the problem can be used to reduce the number of terminals to $2k$ [8] (cf. [23]). For the near-separation variant we consider, it seems unlikely that the linear-programming relaxation has the same nice properties (such as half-integrality) as for the original problem, let alone that the resulting fractional solutions are useful for a reduction in the number of terminals. As one of our main technical ingredients, we therefore develop a combinatorial preprocessing algorithm to reduce the number of terminals to a polynomial in the solution size k plus the size of a given (suboptimal) near-separator S, which will be available via the technique of iterative compression [24] [7, §4]. The preprocessing step is based on concrete reduction rules operating in the graph.

▶ **Theorem 1.2 (★).** *There is a polynomial-time algorithm that, given an instance (G, T, k) of* MWNS *and a multiway near-separator $\hat{S}$ for terminal set T in G, outputs an equivalent instance (G', T', k') such that:*
1. *G' is an induced subgraph of G,*
2. *T' is a subset of T of size $\mathcal{O}(k^5 \cdot |\hat{S}|^4)$, and*
3. *$k' \leq k$.*

Moreover, there is a polynomial-time algorithm that, given a solution S' for (G', T', k'), outputs a solution S of (G, T, k).

Note that the algorithm even runs in *polynomial-time*, and may therefore be a useful ingredient to build a polynomial kernelization for this problem or variations thereof. To obtain this terminal-reduction algorithm, it turns out to be useful to know whether the role of a vertex x in a suboptimal near-separator S can be taken over by $\mathcal{O}(k)$ alternative vertices. If not, then this immediately leads to the conclusion that such a vertex x belongs to any optimal solution to the problem. On the other hand, knowing a small vertex set S_x for which $(S \setminus \{x\}) \cup S_x$ is also a near-separator reveals a lot of structure in the instance which can be exploited by the reduction rule. This usage is similar as the use of the *blocker* [7, §9.1.3] in Thomassé's kernelization algorithm for Feedback Vertex Set [26].

Given a suboptimal near-separator S, we are therefore interested in determining, for a given $x \in S$, whether it is possible to obtain a near-separator S' by replacing x by a set of $\mathcal{O}(k)$ vertices. This is equivalent to finding a near-separator of size $\mathcal{O}(k)$ which avoids the use of vertex x in the graph $G' := G - (S \setminus \{x\})$. Hence this task effectively reduces to finding a solution not containing x in the graph G' for which $\{x\}$ forms a near-separator. We give a polynomial-time 14-approximation for this problem.

▶ **Theorem 1.3 (★).** *There is a polynomial-time algorithm that, given a graph G, terminal set $T \subseteq V(G)$, and a vertex $x \in V(G)$ such that $\{x\}$ is a multiway near-separator for terminal set T in G, outputs a multiway near-separator $S_x \subseteq V(G) \setminus \{x\}$ for T in G such that $|S_x| \leq 14|S_x^*|$, where $S_x^* \not\ni x$ is a smallest multiway near-separator for T in G that avoids x.*

Theorem 1.3 can be compared to a result for the Chordal Deletion problem, where the goal is to delete a minimum number of vertices to break all induced cycles of length at least four (holes). A key step in the polynomial kernelization algorithm for the problem due to Jansen and Pilipczuk is a subroutine ([16, Lemma 1.3]) which, given a graph G and vertex x for which $G - \{x\}$ is chordal, outputs a set of some $\ell \geq 0$ holes pairwise intersecting only in x, together with a vertex set S of size at most 12ℓ not containing x whose removal makes G chordal. Hence S is a 12-approximation for the problem of finding a chordal deletion set which avoids x.

Related work

Apart from the aforementioned work on graph separation problems, the work of Golovach and Thilikos [14] is related to our setting. They consider the problem of removing at most k edges from a graph to split it into exactly t connected components $C_1, \ldots, C_t$ such that C_i has edge-connectivity at least λ_i for a given sequence $(\lambda_1, \ldots, \lambda_t)$. Besides recursive understanding, which only leads to non-uniform FPT classifications, another generic tool for deriving fixed-parameter tractability of separation problems is the treewidth reduction technique by Marx and Razgon [22]. To be able to apply the technique, the number of terminals must be bounded in terms of the parameter k, which is not the case in general.

Even if one uses Theorem 1.2 to bound the number of terminals first, it is not clear if the technique can be applied since the solutions to be preserved are not minimal separators in the graph. Furthermore, successful application of the technique would give double-exponential algorithms at best, due to having to perform dynamic programming on a tree decomposition of width $2^{\Omega(k)}$.

The problem of computing a near-separator avoiding a vertex x is related to the problem of computing an *r-fault tolerant* solution to a vertex deletion problem, which is a solution from which any r vertices may be omitted without invalidating the solution. The computation of r-fault tolerant solutions has been studied for the Feedback Vertex Set problem [2], which has a polynomial-time $\mathcal{O}(r)$-approximation.

The FPT algorithm for Subset Feedback Vertex Set in undirected graphs due to Cygan et al. [9] bears some similarity to ours, in that it also uses reduction rules to bound the number of terminals followed by an algorithm which is exponential in the solution size and the number of terminals. However, Subset Feedback Vertex Set behaves differently from the problem we consider, since in the latter the structures to be hit always involve *pairs* of terminals to be near-separated. On the other hand, a solution to Subset Feedback Vertex Set will reduce the connectivity in the graph to the extent that there will no longer be two internally vertex-disjoint paths between any terminal pair, but also has to ensure that there are no cycles through a single terminal. This leads to significant differences in the approach.

Organization

We begin with short preliminaries with the crucial definitions. We prove our main Theorem 1.1 in Section 3 by assuming Theorem 1.2. Next, in Section 4, we prove Theorem 1.3 by giving a polynomial-time construction of a near-separator avoiding a specific vertex. The proof of Theorem 1.2 is given in Section 5. The proofs of statements marked with (★) are located in the full version [17].

2 Preliminaries

Graphs. We use standard graph-theoretic notation, and we refer the reader to Diestel [11] for any undefined terms. We consider simple unweighted undirected graphs. A graph G has vertex set $V(G)$ and edge set $E(G)$. We use shorthand $n = |V(G)|$ and $m = |E(G)|$. The set $\{1, \ldots, \ell\}$ is denoted by $[\ell]$. The open neighborhood of $v \in V(G)$ is $N_G(v) := \{u \mid \{u, v\} \in E(G)\}$, where we omit the subscript G if it is clear from context. For a vertex set $S \subseteq V(G)$ the open neighborhood of S, denoted $N_G(S)$, is defined as $S := \bigcup_{v \in S} N_G(v) \setminus S$. For $S \subseteq V(G)$, the graph induced by S is denoted by $G[S]$. For two vertices x, y in a graph G, an x-y path is a sequence $(x = v_1, \ldots, v_k = y)$ of vertices such that $\{v_i, v_{i+1}\} \in E(G)$ for all $i \in [k-1]$. Furthermore, the vertices $v_2. \ldots, v_{k-1}$ are called the *internal vertices* of the x-y path. Given a path $P = (v_1, \ldots, v_k)$ and indices $i, j \in [k]$, with $j \geq i$, we use $P[v_i, v_j]$ to denote the subpath of the path P which starts from v_i and ends at v_j. Moreover, we use shorthand $P(v_i, v_j] = P[v_i, v_j] - \{v_i\}, P[v_i, v_j) = P[v_i, v_j] - \{v_j\}$, and $P(v_i, v_j) = P[v_i, v_j] - \{v_i, v_j\}$. Given a p_1-p_k path $P = (p_1, \ldots, p_k)$ and a q_1-q_ℓ path $Q = (q_1, \ldots, q_\ell)$ with $p_k = q_1$ such that P and Q are internally vertex-disjoint, we use $P \cdot Q$ to denote the p_1-q_ℓ path $(p_1, \ldots, p_k, q_2, \ldots, q_\ell)$ obtained by first traversing P and then Q. We say that a path P in G *intersects* a vertex $v_i \in V(G)$ if $v_i \in V(P)$, similarly, for a set $S \subseteq V(G)$, we say that path P intersects S if $V(P) \cap S \neq \emptyset$. For $S \subseteq V(G)$, an x-y path in G is called an S-path if $x, y \in S$. For $S \subseteq V(G)$, cycles C_1, C_2 in G are said to be *S-disjoint* if $V(C_1) \cap V(C_2) \subseteq S$.

We now define a few basic notations about block-cut graphs (for completeness we define the notion of block-cut graph in Definition A.2) that we will use henceforth. Given a graph G with connected components $C_1, \ldots, C_m$, a *rooted block-cut forest* $\mathcal{F}$ of G is a block-cut forest containing block-cut trees $\mathcal{T}_1, \ldots, \mathcal{T}_m$ such that for each $i \in [m]$, the tree $\mathcal{T}_i$ is a block-cut tree of C_i that is rooted at an arbitrary block of $\mathcal{T}_i$. Given a rooted forest $\mathcal{F}$ and a vertex $v \in V(\mathcal{F})$, we use $\text{parent}_{\mathcal{F}}(v)$ to denote the parent of v (if v is a root then $\text{parent}_{\mathcal{F}}(v) = \emptyset$) and $\text{child}_{\mathcal{F}}(v)$ to denote the set containing all children of v (if v is a leaf then $\text{child}_{\mathcal{F}}(v) = \emptyset$). Given a rooted block-cut forest $\mathcal{F}$ of G, and a node d of $\mathcal{F}$, we use $V_G(\mathcal{F}_d)$ to denote the vertices of G occurring in blocks of the subtree rooted at d. Furthermore, we use G_d to denote the graph induced by the vertex set $V_G(\mathcal{F}_d)$, i.e., $G_d := G[V_G(\mathcal{F}_d)]$. We also need the following observations.

▶ **Observation 2.1.** *Let G be a graph, and let B_1, B_2 be two distinct blocks of G such that $V(B_1) \cap V(B_2) = \{v\}$. In the block-cut graph G' of G, it holds that the distance between blocks B_1 and B_2 is two with v as an intermediate vertex.*

▶ **Observation 2.2.** *Consider a graph G, terminal set $T \subseteq V(G)$, and a MWNS $S \subseteq V(G)$ of (G, T). Then each block B of $G - S$ contains at most one terminal.*

Two MWNS instances (G, T, k) and (G', T', k') are said to be *equivalent* if it holds that (G, T, k) is a YES-instance of MWNS if and only if (G', T', k') is a YES-instance of MWNS. An instance (G, T, k) of MWNS is said to be *non-trivial* if $\emptyset$ is not a solution of (G, T, k). A terminal $t \in T$ is said to be *nearly-separated* in G if there does not exist another terminal $t' \in T \setminus \{t\}$ such that there are 2 internally vertex-disjoint t-t' paths in G.

Throughout this manuscript we use Multiway Near-Separator (MWNS) to denote the parameterized version of the multiway near-separator problem, whereas given a graph G and terminal set T we use multiway near-separator (MWNS) to refer to the graph-theoretic concept of nearly-separating a terminal set T in G. Formally, it is defined as follows.

▶ **Definition 2.3** (Multiway near-separator (MWNS)). *Given a graph G and terminal set $T \subseteq V(G)$, a set $S \subseteq V(G)$ is called a multiway near-separator (MWNS) of (G, T) if $S \cap T = \emptyset$ and there does not exist a pair of distinct terminals $t_i, t_j \in T$ such that $G - S$ contains two internally vertex-disjoint t_i-t_j paths.*

▶ **Definition 2.4** (r-redundant MWNS). *Given a graph G and terminal set $T \subseteq V(G)$, a set $S^* \subseteq V(G) \setminus T$ is an r-redundant MWNS of (G, T) if for all $R \subseteq S^*$ with $|R| \leq r$, the set $S^* \setminus R$ is a MWNS of (G, T).*

▶ **Definition 2.5** (x-avoiding MWNS). *Given a graph G, terminal set $T \subseteq V(G)$, and a vertex $x \in V(G)$, a set $S_x \subseteq V(G) \setminus T$ is called an x-avoiding MWNS of (G, T) if $x \notin S_x$ and S_x is a MWNS of (G, T). Among all x-avoiding MWNS of (G, T), one with the minimum cardinality is called a minimum x-avoiding MWNS of (G, T).*

Next, we define T-cycle and give a characterization of MWNS in terms of hitting T-cycles.

▶ **Definition 2.6** (T-cycle and T-cycle on x). *Given a graph G and terminal set $T \subseteq V(G)$, a cycle C in G is called a T-cycle if $|V(C) \cap T| \geq 2$. Moreover, if C also contains a vertex $x \in V(G)$, then C is called a T-cycle on x.*

We now show that several ways of looking at a near-separator are equivalent.

▶ **Proposition 2.7 (★).** *Given a graph G, terminal set $T \subseteq V(G)$, and a non-empty set $S \subseteq V(G) \setminus T$, the following conditions are equivalent:*

1. *For each pair of distinct terminals $t_i, t_j \in T$, the graph $G - S$ does not contain t_i-t_j paths P_1, P_2 which are pairwise internally vertex-disjoint. (Note that P_1 may be identical to P_2 if there are no internal vertices.)*
2. *For each pair of distinct terminals $t_i, t_j \in T$, there is a vertex $v \in V(G) \setminus T$ such that t_i and t_j belong to different connected components of $G - (S \cup \{v\})$.*
3. *The set T is an independent set and $G - S$ does not contain a simple cycle C containing at least two terminals (i.e., a T-cycle).*

Due to space constraints we defer the remaining preliminaries about graphs (including block-cut graphs and important separators) and parameterized algorithms to Appendix A.

3 FPT algorithm for Multiway Near-Separator

In this section we prove Theorem 1.1 assuming Theorem 1.2, which we prove later in Section 5. We use the combination of bounded search trees and iterative compression [7, §3–4] to obtain the FPT algorithm. Towards this, we first present the following structural lemma for a MWNS $S \subseteq V(G)$ of (G, T). It says that in $G - S$, there is a terminal that can simultaneously be separated from *all* other terminals by the removal of a single non-terminal v.

▶ **Lemma 3.1 (★).** *Let (G, T, k) be a non-trivial instance of* MWNS*, and let $S \subseteq V(G) \setminus T$ be a solution. Then there exists a terminal $t \in T$ and a non-terminal vertex $v \in V(G) \setminus T$ such that $S \cup \{v\}$ is a $(t, T \setminus \{t\})$-separator.*

Marx [20] [7, Lemma 8.18] introduced a pushing lemma for Multiway Cut to prove that Multiway Cut is FPT. In the following lemma, we present a pushing lemma for MWNS.

▶ **Lemma 3.2** (Pushing lemma for MWNS (★))**.** *Let (G, T, k) be a non-trivial instance of* MWNS *and let $S \subseteq V(G) \setminus T$ be a solution. Then there exists a terminal $t \in T$ and a solution $S^* \subseteq V(G) \setminus T$ with $|S^*| \leq |S|$ for which one of the following holds:*

1. *there is an important $(t, T \setminus \{t\})$-separator S_t^* of size at most k such that $S_t^* \subseteq S^*$, or*
2. *there is an important $(t, T \setminus \{t\})$-separator S_t of size at most $(k+1)$, and there exists a vertex $v \in S_t$ such that $(S_t \setminus \{v\}) \subseteq S^*$.*

The following lemma forms the heart of the FPT algorithm (Theorem 1.1). It says that there exists an FPT algorithm that can compress a $k+1$-sized MWNS of (G, T) to a k-sized MWNS if (G, T, k) is a YES-instance of MWNS. This is effectively the compression step of the iterative compression technique.

▶ **Lemma 3.3 (★).** *There is an algorithm that, given an instance (G, T, k) of* MWNS *and a set $S_{k+1} \subseteq V(G) \setminus T$ of size $k+1$ such that S_{k+1} is a MWNS of (G, T), runs in time $2^{\mathcal{O}(k \log k)} \cdot n^{\mathcal{O}(1)}$ and outputs a solution of (G, T, k) (of size k) if it exists.*

Given Lemma 3.3, the proof of Theorem 1.1 follows by applying the standard technique of iterative compression. The formal proof can be found in the full version [17].

4 Constructing a near-separator avoiding a specified vertex

In this section we prove Theorem 1.3. Throughout the algorithm, we use the perspective provided by Proposition 2.7 that a MWNS is a set intersecting all T-cycles. Note that since $\{x\}$ is a MWNS for (G, T), the set T must be an independent set. Before presenting the algorithm, we define some notations which we will use during the algorithm.

▶ **Definition 4.1** ($\mathcal{C}(v)$ and $\mathcal{C}_{\geq 1}(v)$)**.** *Given a graph G, terminal set $T \subseteq V(G)$, and a MWNS $\{x\} \subseteq V(G)$ of (G,T), let $\mathcal{F}$ be a rooted block-cut forest of $G - \{x\}$. Let $v \in V(\mathcal{F})$ be a cutvertex. Then we use $\mathcal{C}(v)$ to denote all the grandchildren (cutvertices) of v in the subtree $\mathcal{F}_v$, i.e., $\mathcal{C}(v) := \bigcup_{y \in child_{\mathcal{F}}(v)} child_{\mathcal{F}}(y)$. If v does not have a grandchild then $\mathcal{C}(v) := \emptyset$. We use $\mathcal{C}_{\geq 1}(v) \subseteq \mathcal{C}(v)$ to denote the cutvertices of $\mathcal{C}(v)$ such that for each vertex $c \in \mathcal{C}_{\geq 1}(v)$, the graph $G_c = G[V_G(\mathcal{F}_c)]$ contains a vertex $p \in N_G(x)$ such that there is a c-p path P in G_c which contains at least one terminal, i.e., $|V(P) \cap T| \geq 1$.*

During the construction of an approximate x-avoiding MWNS, we will often make use of Definition 4.1 to keep track of which cutvertices have a pending subgraph attached that can reach a neighbor of x by a simple path containing a terminal. Such subpaths can be combined to form T-cycles. We often use the fact that, in an undirected graph G, it is possible to test in polynomial time whether there is a simple p-q path through a specified vertex t; for example, by constructing a vertex-capacitated flow network in which t has a capacity of 2 and all other vertices a capacity of 1, and testing for a flow from $\{p,q\}$ to $\{t\}$.

Next, we prove some properties about the sets $\mathcal{C}(t)$ and $\mathcal{C}_{\geq 1}(t)$ defined above. We need these properties during the analysis phase (Section 4.2) of the blocker algorithm.

▶ **Proposition 4.2.** *Given a graph G, terminal set $T \subseteq V(G)$, and a MWNS $\{x\} \subseteq V(G)$ of (G,T), let $\mathcal{F}$ be a rooted block-cut forest of $G - \{x\}$. Let $t \in T$ be a cutvertex of $\mathcal{F}$ and let $\mathcal{C}(t)$ be the set of grandchildren of t in the subtree $\mathcal{F}_t$ as defined in Definition 4.1. Then we have $\mathcal{C}(t) \cap T = \emptyset$.*

Proof. Assume for a contradiction that there exists a vertex $t' \in \mathcal{C}(t) \cap T$. First, note that $t' \neq t$, as a cutvertex is present exactly once in a block-cut forest. Thus, we have $t' \in T \setminus \{t\}$. Let $B := \text{parent}_{\mathcal{F}}(t')$. Since $\{t', B\} \in E(\mathcal{F})$, we have $t' \in V(B)$ by definition of block-cut forest. Moreover, as B is the parent of t' and t' is a grandchild of t, we have $\{t, B\} \in E(\mathcal{F})$ and hence $t \in V(B)$. Note that B is a block in $G - \{x\}$ which contains two distinct terminals t, t', a contradiction to Observation 2.2. ◀

▶ **Proposition 4.3.** *Given a graph G, terminal set $T \subseteq V(G)$, and a MWNS $\{x\} \subseteq V(G)$ of (G,T), let $\mathcal{F}$ be a rooted block-cut forest of $G - \{x\}$. Let $t \in T$ be a cutvertex of $\mathcal{F}$ and let $\mathcal{C}_{\geq 1}(t)$ be the subset of grandchildren of t in $\mathcal{F}_t$ defined in Definition 4.1. Let B be a node of $\mathcal{F}_t$ such that the graph $G[V_G(\mathcal{F}_B) \cup \{x\}]$ does not contain a T-cycle. Then the number of cutvertices below B in $\mathcal{F}_B$ which also belong to the set $\mathcal{C}_{\geq 1}(t)$ is at most one, i.e., we have $|V_G(\mathcal{F}_B) \cap C_{\geq 1}(t)| \leq 1$.*

Proof. First of all, note that if B belongs to $\mathcal{C}(t)$ (see Definition 4.1 for the definition of $\mathcal{C}(t)$) or below in the subtree $\mathcal{F}_t$ then the claim trivially holds, because in that case we have $|V_G(\mathcal{F}_B) \cap C_{\geq 1}(t)| \leq 1$. Hence consider the case when either $B \in \text{child}_{\mathcal{F}}(t)$ or $B = t$, and assume for a contradiction that $|V_G(\mathcal{F}_B) \cap C_{\geq 1}(t)| \geq 2$. Let c_1, c_2 be two distinct cutvertices in $V_G(\mathcal{F}_B) \cap C_{\geq 1}(t)$. By definition of the set $\mathcal{C}_{\geq 1}(t)$ and the fact that $c_1, c_2 \in \mathcal{C}_{\geq 1}(t)$, we know that for each $i \in [2]$, the graph G_{c_i} contains a vertex $p_i \in N_G(x)$ such that there is a c_i-p_i path P_i in G_{c_i} containing a terminal t_i. Moreover, since $c_1 \neq c_2$, the paths P_1 and P_2 are vertex disjoint. Next, we do a case distinction based on whether $B \in \text{child}_{\mathcal{F}}(t)$ or $B = t$.

Case 1. When $B \in \text{child}_{\mathcal{F}}(t)$. Since $B \in \text{child}_{\mathcal{F}}(t)$ and by definition (of $\mathcal{C}_{\geq 1}(t)$) c_1, c_2 are grandchildren of t, we have $c_1, c_2 \in \text{child}_{\mathcal{F}}(B)$. Hence, we have $c_1, c_2 \in V_G(B)$. Next, we construct a cycle C in $G[V_G(\mathcal{F}_B) \cup \{x\}]$ as follows. Let $C := \{x, p_1\} \cdot P_1[p_1, c_1] \cdot R_{12}[c_1, c_2] \cdot P_2[c_2, p_2] \cdot \{p_2, x\}$, where R_{12} is a path between cutvertices $c_1, c_2 \in V_G(B)$ inside the block B. Note that the cycle C in $G[V_G(\mathcal{F}_B) \cup \{x\}]$ is simple and contains two distinct terminals $t_1, t_2 \in T$, a contradiction to the fact that there is no T-cycle in $G[V_G(\mathcal{F}_B) \cup \{x\}]$.

Case 2. When $B = t$. For $i \in [2]$, let B_i be the parent of c_i. Note that if $B_1 = B_2$ then similarly to Case 1, we can obtain a T-cycle in $G[V_G(\mathcal{F}_{B_1}) \cup \{x\}]$, which is also a T-cycle in the supergraph $G[V_G(\mathcal{F}_B) \cup \{x\}]$, again a contradiction to the fact that there is no T-cycle in $G[V_G(\mathcal{F}_B) \cup \{x\}]$. Hence assume that $B_1 \neq B_2$. Next, we show that even in this case we can obtain a T-cycle C in $G[V_G(\mathcal{F}_B) \cup \{x\}]$, yielding a contradiction. Indeed, we can use $C := \{x, p_1\} \cdot P_1[p_1, c_1] \cdot R_1[c_1, B] \cdot R_2[B, c_2] \cdot P_2[c_2, p_2] \cdot \{p_2, x\}$, where for $i \in [2]$, the path R_i is a path between vertices $c_i, B \in V_G(B_i)$ inside the block B_i.

Since both the above cases lead to a contradiction, this concludes the proof of Proposition 4.3. ◀

4.1 Algorithm

In this section we present a recursive algorithm Blocker(G, T, x) to construct a set $S_x \subseteq V(G) \setminus (T \cup \{x\})$ such that S_x is a MWNS of (G, T) and $|S_x| \leq 14OPT_x(G, T)$, where $OPT_x(G, T)$ is the cardinality of a minimum x-avoiding MWNS of (G, T). The algorithm effectively takes a graph G, terminal set T, a vertex $x \in V(G) \setminus T$ (such that $\{x\}$ is a MWNS of (G, T)) as input, and computes a vertex set $Z \subseteq V(G) \setminus (T \cup \{x\})$ to hit certain types of T-cycles in G. It combines Z with the result of recursively computing a solution for Blocker$(G - Z, T, x)$. For ease of understanding, we present the algorithm step by step with interleaved comments in italic font, whenever required.

1. If the graph G does not contain a T-cycle on x then return $Z = \emptyset$.
2. Construct a rooted block-cut forest $\mathcal{F}$ of $G - \{x\}$.
3. Choose a deepest node d in the block cut forest $\mathcal{F}$ such that the graph $G[V_G(\mathcal{F}_d) \cup \{x\}]$ contains a T-cycle.
   Note that such a vertex d exists as there is a T-cycle on x in the graph G while a simple cycle in G visits vertices from at most one tree $\mathcal{T} \in \mathcal{F}$.
4. Consider the following cases.
   a. **If d is a cutvertex and $d \notin T$.** Let $Z := \{d\}$. Then return $(Z \cup \text{Blocker}(G - Z, T, x))$.
      In this case, it is easy to observe that the set $Z = \{d\}$ is a MWNS of $(G[V_G(\mathcal{F}_d) \cup \{x\}], T)$ because d is a deepest node satisfying the conditions of Step 3.
   b. **If d is a cutvertex and $d \in T$.** Let $\mathcal{C}_{\geq 1}(d)$ be the subset of grandchildren of d defined using Definition 4.1. Let $Z := \mathcal{C}_{\geq 1}(d)$ and return $(Z \cup \text{Blocker}(G - Z, T, x))$.
      The set Z is a MWNS of $(G[V_G(\mathcal{F}_d) \cup \{x\}], T)$ by our choice of d, definition of the set $\mathcal{C}_{\geq 1}(d)$, the fact that a T-cycle contains at least 2 terminals, and for each block $B \in child_{\mathcal{F}}(d)$ we have $V(B) \cap (T \setminus \{d\}) = \emptyset$ due to Observation 2.2.
   c. **If d is a block.**
      - Let $D_T := d - T$, i.e., $D_T := G[V(d) \setminus T]$. Note that the block d of $G - \{x\}$ contains at most 1 terminal due to Observation 2.2. In the case when $V(d) \cap T = \emptyset$, we have $D_T = d = G[V(d)]$. Let $\mathcal{C}^d := \text{child}_{\mathcal{F}}(d) \setminus T$ and partition $\mathcal{C}^d$ as follows.
        - Let $\mathcal{C}^d_{\geq 2} \subseteq \mathcal{C}^d$ be the set such that for each (cut)vertex $c \in \mathcal{C}^d_{\geq 2}$ the graph $G_c := G[V_G(\mathcal{F}_c)]$ contains a vertex $p \in N_G(x)$ such that there is a c-p path P in G_c which contains at least 2 terminals, i.e., $|V(P) \cap T| \geq 2$.
        - Let $\mathcal{C}^d_1 \subseteq \mathcal{C}^d \setminus \mathcal{C}^d_{\geq 2}$ be the subset of remaining vertices of $\mathcal{C}^d$ such that for each vertex $c \in \mathcal{C}^d_1$ the graph G_c contains a vertex $p \in N_G(x)$ such that there is a c-p path P in G_c which contains 1 terminal, i.e., $|V(P) \cap T| = 1$.
        - Let $\mathcal{C}^d_0 \subseteq \mathcal{C}^d \setminus (\mathcal{C}^d_{\geq 2} \cup \mathcal{C}^d_1)$ be the subset of remaining vertices of $\mathcal{C}^d$ such that for each vertex $c \in \mathcal{C}^d_0$ the graph G_c contains a vertex $p \in N_G(x)$ such that there is a c-p path P in G_c.
        - Let $\mathcal{C}^d_\emptyset := \mathcal{C}^d \setminus (\mathcal{C}^d_{\geq 2} \cup \mathcal{C}^d_1 \cup \mathcal{C}^d_0)$ be the remaining elements of $\mathcal{C}^d$.

- Apply Gallai's theorem ([7, Thm 9.2, Lemma 9.3], cf. [25, Thm 73.1]) on the graph D_T with $Q = \mathcal{C}^d_{\geq 2} \cup \mathcal{C}^d_1$ to obtain a maximum-cardinality family $\mathcal{P}_Q$ of pairwise vertex disjoint Q-paths in D_T, along with a vertex set $Z_1 \subseteq V(D_T)$ of size at most $2|\mathcal{P}_Q|$ such that the graph $D_T - Z_1$ has no Q-path.
- Let $A := \mathcal{C}^d_{\geq 2}$ and $B := \mathcal{C}^d_0 \cup (N_G(x) \cap V(d))$. Next, compute a minimum (A, B)-separator in the graph D_T using Edmonds-Karp algorithm [12] which outputs a vertex set $Z_2 \subseteq V(D_T)$.

Next, we do a case distinction based on whether or not the block d contains a terminal. Note that in the case when d does not contain a terminal then the set $(Z_1 \cup Z_2)$ hits all T-cycles of $G[V_G(\mathcal{F}_d) \cup \{x\}]$. On the other hand, when d contains a terminal there could still be a T-cycle in the graph $G[V_G(\mathcal{F}_d) \cup \{x\}] - (Z_1 \cup Z_2)$ (see the third figure of Figure 1). So our next steps are aimed at hitting those T-cycles (if any) that are not hit by the set $(Z_1 \cup Z_2)$.

- If $V(d) \cap T = \emptyset$, then let $Z_3, Z_4 := \emptyset$.
- Otherwise, there is a unique terminal in block $d \subseteq G - x$ since x is a MWNS for (G, T). Let t be the terminal that belongs to the block d.
  - Let $\mathcal{D}$ be the set containing connected components of $D_T - (Z_1 \cup Z_2)$ that contain at least one neighbor of t, i.e., for each connected component $D \in \mathcal{D}$, we have $V(D) \cap N(t) \neq \emptyset$.
  - Let $\mathcal{D}^* \subseteq \mathcal{D}$ be the set such that for each $D^* \in \mathcal{D}^*$, the connected component D^* contains a vertex from $Q = (\mathcal{C}^d_{\geq 2} \cup \mathcal{C}^d_1)$. More precisely, we have $|V(D^*) \cap Q| = 1$ for each D^* as the set Z_1 is hitting all Q-paths.
  - Let $V(\mathcal{D}^*) := \bigcup_{D^* \in \mathcal{D}^*} V(D^*)$ and define $Z_3 := (\mathcal{C}^d_{\geq 2} \cup \mathcal{C}^d_1) \cap V(\mathcal{D}^*)$.
    Note that in the case when d contains a terminal t and $t \in child_{\mathcal{F}}(d)$, the way we have defined $\mathcal{C}^d$, it does not contain t. Hence, when $t \in child_{\mathcal{F}}(d)$ and the graph G_t has a vertex $p \in N_G(x)$ such that there is a t-p path P in G_t that contains at least one terminal other than t, i.e, $|V(P) \cap (T \setminus \{t\})| \geq 1$, there could still be T-cycles in $G[V_G(\mathcal{F}_d)] - \bigcup_{i=1}^3 Z_i$ (see the last figure of Figure 1). So our next step is to hit all T-cycles (if any) containing the t-p path P. Recall $\mathcal{C}_{\geq 1}(t)$ from Definition 4.1.
    - If $t \in \text{child}_{\mathcal{F}}(d)$ and the graph G_t has a vertex $p \in N_G(x)$ such that there is a t-p path P in G_t with $|V(P) \cap T| \geq 2$, then let $Z_4 := \mathcal{C}_{\geq 1}(t)$.
    - Otherwise, define $Z_4 := \emptyset$.

Finally, we try to break any interaction between vertices of $V_G(\mathcal{F}_d)$ and vertices of $V_G(\mathcal{F}) \setminus V_G(\mathcal{F}_d)$ by adding the parent of d into the hitting set. But note that in the case when $parent_{\mathcal{F}}(d) \in T$, we can not add it to the hitting set $Z \subseteq V(G) \setminus (T \cup \{x\})$.

- If $\text{parent}_{\mathcal{F}}(d) \in T$, then let $Z_5 := \emptyset$.
- Otherwise, $Z_5 := \text{parent}_{\mathcal{F}}(d)$.

Let $Z := \bigcup_{i=1}^5 Z_i$ and return$(Z \cup \text{Blocker}(G - Z, T, x))$.

This concludes the description of the algorithm. Summarizing, its main structure is to define a vertex set $Z \subseteq V(G) \setminus (T \cup \{x\})$ to break T-cycles which are *lowest* in the block-cut tree, include that set Z in the approximate solution, and complete the solution by recursively solving the problem on $G - Z$.

4.2 Analysis

The following lemma forms the heart of Theorem 1.3. It says that if the above procedure adds a set Z during the construction of the approximate solution S_x, then the optimum value of the remaining instance decreases by at least $\frac{|Z|}{14}$.

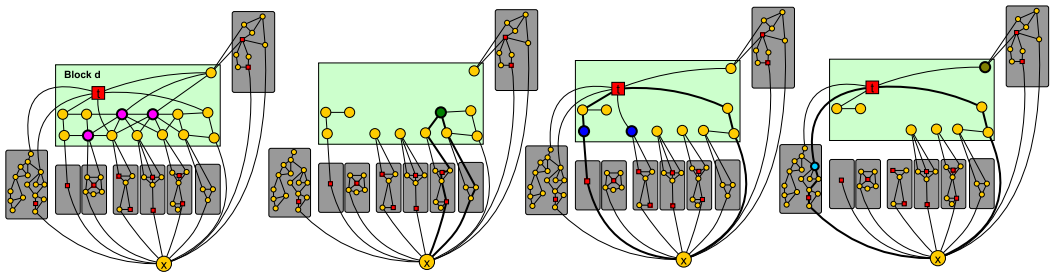

Figure 1 Illustration of Step 4c. The leftmost figure shows the original graph G where terminals are represented by red squares and the pink vertices of block d represent the vertices of Z_1. The second figure shows the construction of the set Z_2 in the graph $G - (Z_1 \cup \{t\})$ where the green vertex represents the vertex of Z_2. It also shows a T-cycle represented by thick edges. The third figure shows a T-cycle (represented by thick edges) which appears after putting the terminal t back and the blue vertices represent the vertices of Z_3. The last figure illustrates the construction of the set Z_4 and Z_5 where the cyan vertex and olive vertex represent the vertex of Z_4 and Z_5, respectively.

▶ **Lemma 4.4.** *If a single iteration of the algorithm on input (G, T, x) yields the set Z, then $\mathrm{OPT}_x(G - Z, T) \leq \mathrm{OPT}_x(G, T) - \frac{|Z|}{14}$, where $\mathrm{OPT}_x(G, T)$ and $\mathrm{OPT}_x(G - Z, T)$ are the cardinalities of a minimum x-avoiding MWNS of (G, T) and $(G - Z, T)$, respectively.*

Proof. Let $S_x^* \subseteq V(G)$ be a minimum x-avoiding MWNS of (G, T). If the algorithm stops before Step (3) then $Z = \emptyset$. Hence the inequality of the lemma trivially holds. Therefore assume that the algorithm reaches Step 4. Let d be the deepest node selected by the algorithm in Step (3) to compute Z. Let $\hat{S} := S_x^* \setminus V_G(\mathcal{F}_d)$. Note that $\hat{S} \cap (T \cup \{x\}) = \emptyset$. Next, we observe the following properties about the set Z computed in Step 4.

▷ Claim 4.5 (★). Suppose the algorithm reaches Step 4 and computes the set $Z \subseteq V_G(\mathcal{F}_d) \setminus T$. Then Z is a MWNS of $(G_d + x, T)$, where $G_d + x = G[V_G(\mathcal{F}_d) \cup \{x\}]$.

Proof sketch. By choice of d, each T-cycle of $G_d + x$ contains a vertex from block d, or the cutvertex d itself. Since x is a MWNS for (G, T), each T-cycle also contains x. The sets $Z_1, \ldots, Z_4$ added to Z in the algorithm ensure different types of T-cycles of $G_d + x$ are broken: Z_1 covers all T-cycles consisting of two paths between x and d, each containing at least one terminal; Z_2 covers T-cycles consisting of one path between x and d containing two or more terminals, and another such path containing no terminals; the sets Z_3 and Z_4 cover T-cycles that go through a terminal in d (if there is one), as explained in the algorithm. ◁

▷ Claim 4.6 (★). Suppose the algorithm reaches Step 4 and computes the set $Z \subseteq V_G(\mathcal{F}_d) \setminus T$. Then any T-cycle in $G - (Z \cup \hat{S})$ contains a vertex of $V_G(\mathcal{F}_d)$ (i.e., a vertex from G_d) and a vertex from $V_G(\mathcal{F}) \setminus (V_G(\mathcal{F}_d))$ (i.e., a vertex outside $G_d + x$).

Proof sketch. Since $\hat{S}$ contains all vertices of the solution S_x^* except those occurring in a block of the subtree $\mathcal{F}_d$ of the block-cut tree, any T-cycle disjoint from $\mathcal{S}$ was intersected by S_x^* in a vertex of $V_G(\mathcal{F}_d)$ and therefore uses a vertex of the latter set. On the other hand, since the previous claim shows that Z hits all the T-cycles which live in $G_d + x$, a T-cycle disjoint from Z has to use a vertex outside $V_G(\mathcal{F}_d)$. ◁

▷ Claim 4.7 (★). Suppose the algorithm reaches Step 4 and computes the set $Z \subseteq V_G(\mathcal{F}_d) \setminus T$. Then any minimum x-avoiding MWNS S_x^* of (G, T) satisfies $|S_x^* \cap V_G(\mathcal{F}_d)| \geq \max\{1, \frac{|Z|-2}{6}\}$.

Proof sketch. Any x-avoiding MWNS S_x^* contains a vertex from $V_G(\mathcal{F}_d)$ because there is a T-cycle in $G_d + x$, by choice of d, which explains why the intersection is nonempty. The fact that S_x^* contains at least $\frac{|Z|-2}{6}$ vertices from $V_G(\mathcal{F}_d)$ can be seen as follows. We have $|Z_4|, |Z_5| \leq 1$ by definition, so the largest Z_i of Z_1, Z_2, Z_3 has at least $\frac{|Z|-2}{3}$ vertices. Any solution contains at least $|Z_i|/2$ vertices from $V_G(\mathcal{F}_d)$ because the covering/packing duality for the three types of separators used to define Z_1, Z_2, Z_3 ensures, for each of these sets, that to hit all the T-cycles of the corresponding form, at least $|Z_i|/2$ vertices are needed. For example, each Q-path P in the family $\mathcal{P}_Q$ obtained during the construction of Z_1 yields a T-cycle when combined with paths from endpoints of P (which are cutvertices in Q) to neighbors of x in two different subtrees of the block-cut forest $\mathcal{F}$, showing that $S_x^* \cap V_G(\mathcal{F}_d) \geq |\mathcal{P}_Q| \geq |Z_1|/2$. ◁

Next, we show that if a minimum x-avoiding MWNS S_x^* of (G, T) contains exactly 1 vertex from the subtree $\mathcal{F}_d$ then the set $\hat{S} := S_x^* \setminus V_G(\mathcal{F}_d)$ is a MWNS of $(G - Z, T)$.

▷ Claim 4.8 (★). If $|S_x^* \cap V_G(\mathcal{F}_d)| = 1$ then $\hat{S} = S_x^* \setminus V_G(\mathcal{F}_d)$ is a MWNS of $(G - Z, T)$.

Proof sketch. If $|S_x^* \cap V_G(\mathcal{F}_d)| = 1$, then from the T-cycle in $G_d + x$ which we know to exist, the set S_x^* contains at most one vertex. In the most crucial case that the d is a block whose parent is a terminal, this means that S_x^* cannot break all paths from x through blocks in the subtree $\mathcal{F}_d$ to the parent of d. The only types of connections that the set Z does not break, and which can be part of T-cycles in G, can be shown to be precisely paths from a neighbor of x to d in G_d that do not contain a terminal. But if $|S_x^* \cap V_G(\mathcal{F}_d)| = 1$, then S_x^* does not break all such paths either. By a rerouting argument, this allows us to show that updating S^* by replacing $S_x^* \cap V_G(\mathcal{F}_d)$ with Z gives a valid x-avoiding MWNS, which is equivalent to saying that $\hat{S}$ is a MWNS of $(G - Z, T)$. ◁

The following claim shows that if the set $\hat{S}$ is not a MWNS of $(G - Z, T)$, then we can find a vertex $\hat{c}$ such that the set obtained after adding $\hat{c}$ to the set $\hat{S}$ is a MWNS of $(G - Z, T)$.

▷ Claim 4.9 (★). If $\hat{S}$ is not a MWNS of $(G - Z, T)$ then there exists a vertex $\hat{c} \in V(G) \setminus (T \cup \{x\})$ such that $\hat{S} \cup \{\hat{c}\}$ is a MWNS of $(G - Z, T)$.

Proof sketch. The proof consists of a delicate argument which essentially says that this situation only happens when d is a block whose parent is a terminal, and there is a cutvertex $\hat{c}$ close to block d in the block-cut forest which combines with Z to break all paths in $G - \{x\}$ from $N_G(x) \cap V_G(\mathcal{F}_d)$ to vertices of $N_G(x) \setminus V_G(\mathcal{F}_d)$, and therefore breaks all T-cycles intersecting $V_G(\mathcal{F}_d)$. ◁

The proof of Lemma 4.4 follows from the preceding statements by formula manipulation: whenever the approximation algorithm chooses a set Z, an optimal solution chooses at least $|Z|/14$ vertices from $V_G(\mathcal{F}_d)$. The remainder of the proof is given in the full version [17]. ◀

Using Lemma 4.4, an easy induction shows that the algorithm indeed computes a 14-approximation. As each iteration can be implemented in polynomial time, this leads to a proof of Theorem 1.3. The details are given in the full version [17].

5 Bounding the number of terminals

Our main goal in this section is to prove Theorem 1.2. Intuitively, there are two distinct ways in which a connected component admitting a small solution can contain a large number of terminals: either there can be a star-like structure of blocks containing a terminal joined at a single cutvertex, or there exists a long path in the block-cut tree containing many blocks with a terminal. After applying reduction rules to attack both kinds of situations, we give a final reduction rule to bound the number of relevant connected components that contain a terminal, which will lead to the desired bound on the number of terminals. Due to space constraints, the safeness of the reduction rules we present here is deferred to the full version [17].

▶ **Reduction Rule 1.** *Let (G, T, k) be an instance of* MWNS*, and let $t \in T$ be a terminal such that for any other terminal $t' \in T \setminus \{t\}$, there do not exist 2 internally vertex-disjoint t-t' paths in G, i.e., the terminal t is nearly-separated. Then remove t from the set T. The new instance is $(G, T \setminus \{t\}, k)$.*

Next, we have the following general reduction rule.

▶ **Reduction Rule 2.** *Let (G, T, k) be an instance of* MWNS*. Suppose there exist 2 non-terminal vertices $x, y \in V(G) \setminus T$ such that $G - \{x, y\}$ has a connected component D satisfying the following conditions:*

(a) *$|V(D) \cap T| \geq 3$,*

(b) *the graph $G[V(D) \cup \{x, y\}]$ has no T-cycle, and*

(c) *there is an x-y path in $G[V(D) \cup \{x, y\}]$ containing distinct terminals $t_1, t_2 \in T$.*

Then turn any terminal of D which is not t_1, t_2 into a non-terminal. Formally, let $t \in V(D) \cap (T \setminus \{t_1, t_2\})$, then $(G, T \setminus \{t\}, k)$ is a new instance of MWNS*.*

Next, we show that given an instance (G, T, k) of MWNS and a 1-redundant MWNS $S^* \subseteq V(G) \setminus T$ of (G, T), in polynomial-time we can obtain an equivalent instance (G, T', k) such that the number of connected components of $G - S^*$ which contain at least one terminal from T' is bounded by $\mathcal{O}(|S^*|^2 \cdot k)$. Towards this, we apply the following marking scheme with respect to the set S^* to mark $\mathcal{O}(|S^*|^2 \cdot k)$ connected components of $G - S^*$.

▶ **Marking Scheme 1.** *Let (G, T, k) be an instance of* MWNS*, and let $S^* \subseteq V(G)$ be a 1-redundant MWNS of (G, T). For each pair of distinct vertices $x, y \in S^*$, we greedily mark $(k+2)$ connected components $C^{x,y}_{i_1}, \ldots, C^{x,y}_{i_{k+2}}$ of $G - S^*$ such that for each $m \in [k+2]$, the connected component $C^{x,y}_{i_m}$ contains two vertices (not necessarily distinct) $p_m \in N_G(x)$ and $q_m \in N_G(y)$ such that there is a p_m-q_m path P_m (possibly of length 0) inside the connected component $C^{x,y}_m$ which contains at least one terminal, i.e., we have $|V(P_m) \cap T| \geq 1$. If there are fewer than $k+2$ such components, we simply mark all of them.*

We have the following reduction rule based on the above marking scheme. It requires a 1-redundant MWNS to be known. In the context of Theorem 1.2, where we are given a MWNS S of (G, T, k), we can exploit Theorem 1.3 to obtain a 1-redundant MWNS of size $\mathcal{O}(|S| \cdot k)$ in polynomial time: for each $x \in S$, if an x-avoiding MWNS S_x in $G - (S \setminus \{x\})$ has size more than $14k$ then all solutions of (G, T, k) contain x and we may safely remove x and decrease k; otherwise, we can add S_x to S to ensure all T-cycles intersecting x are hit twice, without blowing up the size of S too much. The details are given in the full version [17].

▶ **Reduction Rule 3.** *Let (G, T, k) be an instance of* MWNS*, and let $S^* \subseteq V(G) \setminus T$ be a 1-redundant MWNS of (G, T). Let $\mathcal{C}_M$ be the set of connected components of $G - S^*$ marked by Marking Scheme 1 with respect to the set S^*. Let $T' := T \cap (\bigcup_{C \in \mathcal{C}_M} V(C))$. If $T \setminus T' \neq \emptyset$ then we convert the terminals of $T \setminus T'$ to non-terminals. The new instance is (G, T', k).*

A delicate analysis shows that applying these reduction rules indeed leads to an equivalent instance with the desired bound on the number of terminals, which leads to a proof of Theorem 1.2. The details are given in the full version [17].

6 Conclusions

In this paper we initiated the study of the Multiway Near-Separator problem, a generalization of Multiway Cut focused on reducing the connectivity between each pair of terminals. We developed reduction rules to reduce the number of terminals in an instance, aided by a constant-factor approximation algorithm for the problem of finding a near-separator avoiding a given vertex x in an instance for which $\{x\}$ is a near-separator. Our work leads to several follow-up questions. First of all, one could consider extending the notion of near-separation to allow for larger (but bounded) connectivity between terminal pairs in the resulting instance, for example by requiring that in the graph $G - S$, for each pair of distinct terminals t_i, t_j there is a vertex set of size at most c whose removal separates t_i and t_j. The setting we considered here is that of $c = 1$, which allows us to understand the structure of the problem based on the block-cut forest of $G - S$. For $c = 2$ it may be feasible to do an analysis based on the decomposition of $G - S$ into triconnected components, but for larger values of c the structure may become significantly more complicated. One could also consider a generic setting (see [1]) where for each pair of terminals t_i, t_j, some threshold $f(t_i, t_j) = f(t_j, t_i)$ is specified such that in $G - S$ there should be a set of at most $f(t_i, t_j)$ nonterminals whose removal separates t_i from t_j. Note that if the values of f are allowed to be arbitrarily large, this generalizes Multicut. Is the resulting problem fixed-parameter tractable parameterized by k?

Another direction for future work lies in the development of a polynomial kernel. For the Multiway Cut problem with delectable terminals, as well as the setting with undeletable but constantly many terminals, a polynomial kernel is known based on matroid techniques [18]. Does the variant of Multiway Near-Separator with deletable terminals admit a polynomial kernel?

References

1 Siddharth Barman and Shuchi Chawla. Region growing for multi-route cuts. In Moses Charikar, editor, *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2010, Austin, Texas, USA, January 17-19, 2010*, pages 404–418. SIAM, 2010. doi:10.1137/1.9781611973075.34.

2 Václav Blazej, Pratibha Choudhary, Dusan Knop, Jan Matyás Kristan, Ondrej Suchý, and Tomás Valla. Constant factor approximation for tracking paths and fault tolerant feedback vertex set. In Jochen Könemann and Britta Peis, editors, *Approximation and Online Algorithms - 19th International Workshop, WAOA 2021, Lisbon, Portugal, September 6-10, 2021, Revised Selected Papers*, volume 12982 of *Lecture Notes in Computer Science*, pages 23–38. Springer, 2021. doi:10.1007/978-3-030-92702-8_2.

3 Nicolas Bousquet, Jean Daligault, and Stéphan Thomassé. Multicut is FPT. *SIAM J. Comput.*, 47(1):166–207, 2018. doi:10.1137/140961808.

4 Karl Bringmann, Danny Hermelin, Matthias Mnich, and Erik Jan van Leeuwen. Parameterized complexity dichotomy for steiner multicut. *J. Comput. Syst. Sci.*, 82(6):1020–1043, 2016. doi:10.1016/j.jcss.2016.03.003.

5 Jianer Chen, Yang Liu, and Songjian Lu. An improved parameterized algorithm for the minimum node multiway cut problem. *Algorithmica*, 55(1):1–13, 2009. doi:10.1007/s00453-007-9130-6.

6 Rajesh Chitnis, Marek Cygan, MohammadTaghi Hajiaghayi, Marcin Pilipczuk, and Michal Pilipczuk. Designing FPT algorithms for cut problems using randomized contractions. *SIAM J. Comput.*, 45(4):1171–1229, 2016. doi:10.1137/15M1032077.

7 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.

8 Marek Cygan, Marcin Pilipczuk, Michal Pilipczuk, and Jakub Onufry Wojtaszczyk. On multiway cut parameterized above lower bounds. *ACM Trans. Comput. Theory*, 5(1):3:1–3:11, 2013. doi:10.1145/2462896.2462899.

9 Marek Cygan, Marcin Pilipczuk, Michal Pilipczuk, and Jakub Onufry Wojtaszczyk. Subset feedback vertex set is fixed-parameter tractable. *SIAM J. Discret. Math.*, 27(1):290–309, 2013. doi:10.1137/110843071.

10 E. Dahlhaus, D. S. Johnson, C. H. Papadimitriou, P. D. Seymour, and M. Yannakakis. The complexity of multiway cuts (extended abstract). In *Proceedings of the Twenty-Fourth Annual ACM Symposium on Theory of Computing*, STOC '92, pages 241–251, New York, NY, USA, 1992. Association for Computing Machinery. doi:10.1145/129712.129736.

11 Reinhard Diestel. *Graph Theory, 5th Edition*, volume 173 of *Graduate texts in mathematics*. Springer, 2017. doi:10.1007/978-3-662-53622-3.

12 Jack Edmonds and Richard M. Karp. Theoretical improvements in algorithmic efficiency for network flow problems. *J. ACM*, 19(2):248–264, April 1972. doi:10.1145/321694.321699.

13 L. R. Ford and D. R. Fulkerson. Maximal flow through a network. *Canadian Journal of Mathematics*, 8:399–404, 1956. doi:10.4153/CJM-1956-045-5.

14 Petr A. Golovach and Dimitrios M. Thilikos. Clustering to given connectivities. In Bart M. P. Jansen and Jan Arne Telle, editors, *14th International Symposium on Parameterized and Exact Computation, IPEC 2019, September 11-13, 2019, Munich, Germany*, volume 148 of *LIPIcs*, pages 18:1–18:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPIcs.IPEC.2019.18.

15 Sylvain Guillemot. FPT algorithms for path-transversal and cycle-transversal problems. *Discret. Optim.*, 8(1):61–71, 2011. doi:10.1016/j.disopt.2010.05.003.

16 Bart M. P. Jansen and Marcin Pilipczuk. Approximation and kernelization for chordal vertex deletion. *SIAM J. Discret. Math.*, 32(3):2258–2301, 2018. doi:10.1137/17M112035X.

17 Bart M. P. Jansen and Shivesh K. Roy. On the parameterized complexity of multiway near-separator, 2023. arXiv:2310.04332.

18 Stefan Kratsch and Magnus Wahlström. Representative sets and irrelevant vertices: New tools for kernelization. *J. ACM*, 67(3):16:1–16:50, 2020. doi:10.1145/3390887.

19 Daniel Lokshtanov, M. S. Ramanujan, Saket Saurabh, and Meirav Zehavi. Reducing CMSO model checking to highly connected graphs. In Ioannis Chatzigiannakis, Christos Kaklamanis, Dániel Marx, and Donald Sannella, editors, *45th International Colloquium on Automata, Languages, and Programming, ICALP 2018, July 9-13, 2018, Prague, Czech Republic*, volume 107 of *LIPIcs*, pages 135:1–135:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPIcs.ICALP.2018.135.

20 Dániel Marx. Parameterized graph separation problems. *Theor. Comput. Sci.*, 351(3):394–406, 2006. doi:10.1016/j.tcs.2005.10.007.

21 Dániel Marx, Barry O'Sullivan, and Igor Razgon. Finding small separators in linear time via treewidth reduction. *ACM Trans. Algorithms*, 9(4):30:1–30:35, 2013. doi:10.1145/2500119.

22 Dániel Marx and Igor Razgon. Fixed-parameter tractability of multicut parameterized by the size of the cutset. *SIAM J. Comput.*, 43(2):355–388, 2014. doi:10.1137/110855247.

23 Igor Razgon. Large isolating cuts shrink the multiway cut. *CoRR*, abs/1104.5361, 2011. arXiv:1104.5361.

24 Bruce A. Reed, Kaleigh Smith, and Adrian Vetta. Finding odd cycle transversals. *Oper. Res. Lett.*, 32(4):299–301, 2004. doi:10.1016/j.orl.2003.10.009.

25 A. Schrijver. *Combinatorial Optimization - Polyhedra and Efficiency.* Springer, 2003.

26 Stéphan Thomassé. A $4k^2$ kernel for feedback vertex set. *ACM Trans. Algorithms*, 6(2):32:1–32:8, 2010. doi:10.1145/1721837.1721848.

27 Mingyu Xiao. Simple and improved parameterized algorithms for multiterminal cuts. *Theory Comput. Syst.*, 46(4):723–736, 2010. doi:10.1007/s00224-009-9215-5.

A Additional preliminaries

Graphs. An x-y path P and a q-r path Q are said to be internally vertex-disjoint if they do not share a common internal-vertex, i.e., we have $(V(P) \setminus \{x, y\}) \cap (V(Q) \setminus \{q, r\}) = \emptyset$. Given $X, Y \subseteq V(G)$, a path $(v_1, \ldots, v_k)$ in G is called an $X - Y$ path if $v_1 \in X$ and $v_k \in Y$.

▶ **Theorem A.1** (Menger's theorem, [11], Theorem 3.3.1)**.** *Let G be a graph and $X, Y \subseteq V(G)$ be subsets of vertices such that $X \cap Y = \emptyset$ and there does not exist an edge $\{x, y\} \in E(G)$ for any $x \in X$ and $y \in Y$. Then the minimum number of vertices separating X from Y is equal to the maximum number of vertex-disjoint $X - Y$ paths in G.*

A cutvertex in a graph is a vertex v whose removal increases the number of connected components. A graph is 2-connected if it has at least three vertices and does not contain any cutvertex. A *block* of a graph G is a maximal connected subgraph B of G such that B does not have a cutvertex. Each block of G is either a 2-connected subgraph of G, a single edge, or an isolated vertex.

▶ **Definition A.2** (Block-cut graph, [11], §3.1)**.** *Given a graph G, let A be the set of cutvertices of G, and let $\mathcal{B}$ be the set of its blocks. The* block-cut graph *G' of G is the bipartite graph with partite sets A and $\mathcal{B}$, and for each cut-vertex $a \in A$, for each block $B \in \mathcal{B}$, there is an edge $\{a, B\} \in E(G')$ if $a \in V(B)$.*

We also need the following simple but useful properties of block-cut graphs.

▶ **Lemma A.3** ([11], Lemma 3.1.4)**.** *The block-cut graph of a connected graph is a tree.*

▶ **Observation A.4.** *Consider an edge e of the block-cut tree $\mathcal{T}$ of a connected graph G, let v be the unique cutvertex incident on e, let $\mathcal{T}_1, \mathcal{T}_2$ be the two trees of $\mathcal{T} - \{e\}$, and let $Y_i := V_G(\mathcal{T}_i)$ be the vertices of G occurring in blocks of $\mathcal{T}_i$, for $i \in [2]$. Then all paths from a vertex of $Y_1 \setminus \{v\}$ to a vertex of $Y_2 \setminus \{v\}$ in G intersect v.*

▶ **Observation A.5.** *Let G be a graph and $T \subseteq V(G)$ be a set of terminals such that $V(G) \setminus T \neq \emptyset$. If no block of G contains two or more terminals, then for each pair $t_i, t_j \in T$ of distinct terminals there exists a vertex $v \in V(G) \setminus T$ such that v_i and v_j belong to different connected components of $G - \{v\}$.*

▶ **Observation A.6.** *For any positive integer ℓ, if there are ℓ cycles on x which are $(T \cup \{x\})$-disjoint in graph G, then any x-avoiding MWNS of (G, T) contains at least ℓ vertices: at least one distinct non-terminal from each cycle.*

▶ **Proposition A.7.** *Let G, be a graph, B a block in G, and consider three distinct vertices $p, q, t \in V(B)$. There is a p-t path P and q-t path Q inside block B such that $V(P) \cap V(Q) = \{t\}$.*

Proof. Since B is a block containing at least three vertices, it is a 2-connected graph. We first add a new vertex $v_{p,q}$ and edges $\{v_{pq}, p\}$, $\{v_{pq}, q\}$ to block B to obtain a new graph B'. Observe that B' is still a 2-connected graph, as the above modification can be seen as adding a new B-path v, v_{pq}, q to B and adding a B-path to a 2-connected graph results in a 2-connected graph (see [11, Proposition 3.1.1]). Next, we apply Menger's theorem with $X = \{v_{pq}\}$ and $Y = \{t\}$ in the (2-connected) graph B', which ensures that there are 2 internally vertex-disjoint v_{pq}-t paths P' and Q' in B'. By construction of B' and the fact that P' and Q' are internally vertex-disjoint v_{pq}-t paths in B', we know that one of the v_{pq}-t paths (assume w.l.o.g. P') contains vertex p of block B, whereas the other v_{pq}-t path Q' contains vertex q of block B. Hence the paths $P = P'[p, t]$ and $Q = Q'[q, t]$ are the desired paths with $V(P) \cap V(Q) = \{t\}$. ◀

This following observation follows from Proposition A.7, as explained below.

▶ **Observation A.8.** *Let $\mathcal{F}$ be the block-cut forest of a graph G. Suppose there is an x-y path $\mathcal{P}$ in $\mathcal{F}$ between cutvertices x, y of $\mathcal{F}$. Then for any distinct vertices $v_1, v_2 \in V(G)$ from 2 distinct blocks on $\mathcal{P}$, the graph G has an x-y path P_{12} through v_1, v_2, i.e., $v_1, v_2 \in V(P_{12})$.*

We can construct the desired path P_{12} by concatenating paths inside each block B on the x-y path in $\mathcal{F}$, where each path connects the cutvertex p connecting to the previous block, with the cutvertex q connecting through the next block. If a forced v_i is chosen from block B, we can use Proposition A.7 to obtain a p-q path through $t = v_i$.

Parameterized algorithms. A *parameterized problem* L is a subset of $\Sigma^* \times \mathbb{N}$, where Σ is a finite alphabet. A parameterized problem $L \subseteq \Sigma^* \times \mathbb{N}$ is called *fixed parameter tractable* (FPT) if there exists an algorithm which for every input $(x, k) \in \Sigma^* \times \mathbb{N}$ correctly decides whether $(x, k) \in L$ in $f(k) \cdot |x|^{\mathcal{O}(1)}$ time, where $f \colon \mathbb{N} \to \mathbb{N}$ is a computable function. We refer to [7] for more background on parameterized algorithms.

Given a graph G and $X, Y \subseteq V(G)$, a set $S \subseteq V(G)$ is called an (X, Y)-*separator* if there is no x-y path in $G - S$ for any $x \in X \setminus S$ and $y \in Y \setminus S$. The notion of important separator was defined by Marx [20] to obtain an FPT algorithm for Multiway Cut. He also derived several useful properties.

▶ **Definition A.9** (Important separator, [7], Definition 8.49)**.** *Let G be an undirected graph, let $X, Y \subseteq V(G)$ be two sets of vertices, and let $V^\infty \subseteq V(G)$ be a set of undeletable vertices. Let $S \subseteq V(G) \setminus V^\infty(G)$ be an (X, Y)-separator and let R be the set of vertices reachable from $X \setminus S$ in $G - S$. We say that S is an* important (X, Y)-separator *if it is inclusion-wise* minimal *and there is no (X, Y)-separator $S' \subseteq V(G) \setminus V^\infty(G)$ with $|S'| \leq |S|$ such that $R \subset R'$, where R' is the set of vertices reachable from $X \setminus S'$ in $G - S'$.*

▶ **Proposition A.10** ([7], Proposition 8.50)**.** *Let G be an undirected graph and $X, Y \subseteq V(G)$ be two sets of vertices, and let $V^\infty \subseteq V(G)$ be a set of undeletable vertices. Let $\hat{S} \subseteq V(G) \setminus V^\infty(G)$ be an (X, Y)-separator and let $\hat{R}$ be the set of vertices reachable from $X \setminus \hat{S}$ in $G - \hat{S}$. Then there is an important (X, Y)-separator $S' = N_G(R') \subseteq V(G) \setminus V^\infty(G)$ such that $|S'| \leq |\hat{S}|$ and $\hat{R} \subseteq R'$.*

▶ **Theorem A.11** ([7], Theorem 8.51)**.** *Let $X, Y \subseteq V(G)$ be two sets of vertices in an undirected graph G, let $k \geq 0$ be an integer, and let $\mathcal{S}_k$ be the set of all (X, Y)-important separators of size at most k. Then $|\mathcal{S}_k| \leq 4^k$ and $\mathcal{S}_k$ can be constructed in time $\mathcal{O}(|\mathcal{S}_k| \cdot k^2 \cdot (n + m))$.*

B Hardness proof for Multiway Near-Separator

▶ **Lemma B.1.** *Multiway Near-Separator is NP-hard.*

Proof. We give a polynomial-time reduction from Multiway Separator to mwns. The Multiway Separator problem is formally defined as follows.

Multiway Separator (mws)	**Parameter:** k
Input: An undirected graph G, terminal set $T \subseteq V(G)$, and a positive integer k.	
Question: Is there a set $S \subseteq V(G) \setminus T$ with $\lvert S\rvert \leq k$ such that there does not exist a pair of distinct terminals $t_i, t_j \in T$ for which there is a t_i-t_j path in $G - S$?	

Multiway Separator is NP-hard [10]. Given an instance (G, T, k) of mws we describe the construction of an instance (G', T, k) of mwns; it will be easy to see that it can be carried out in polynomial time. Consider an arbitrary ordering $t_1, t_2, \ldots, t_{|T|}$ of the set T. We construct the graph G' such that (G', T, k) is a YES-instance of mwns if and only if (G, T, k) is a YES-instance of mws.

We begin with $G' := G$. Then for each $i \in [|T| - 1]$, we create a vertex w_i in G', and insert edges $\{t_i, w_i\}$ and $\{w_i, t_{i+1}\}$ in G'. This completes the construction of G'.

Next, we prove that (G, T, k) is a YES-instance of mws if and only if (G', T, k) is a YES-instance of mwns. In the forward direction, assume that (G, T, k) is a YES-instance of mws, and let $S \subseteq V(G) \setminus T$ be a mws of (G, T, k). Note that by definition of mws, for each pair of distinct terminals $t_i, t_j \in T$, there is no t_i-t_j path in $G - S$. Since the transformation into G' consists of adding degree-2 vertices connecting consecutive terminals, any pair of distinct terminals $t_i, t_j \in T$ with $i < j$ can be separated in $G' - S$ by removing the vertex w_i. Hence the same set $S \subseteq V(G') \setminus T$ is also a mwns of (G', T, k).

In the reverse direction, assume that (G', T, k) is a YES-instance of mwns and let $S' \subseteq V(G') \setminus T$ be a solution. Let $W := \bigcup_{i \in [|T|-1]} \{w_i\}$ be the set of newly added vertices in G'. In the case when $S' \cap W = \emptyset$, it is easy to see that the same set S' is also a mws of (G, T, k): because if S is not a mws of (G, T, k) then it implies that there is a pair of distinct terminals $t_i, t_j \in T$ connected by a t_i-t_j path say P in $G - S$. By construction of G', there is also a unique t_i-t_j path say P' in $G'[W \cup T]$. Note that the paths P and P' are T-disjoint, and neither P nor P' intersects the set S', a contradiction to the fact that S' is a solution of (G', T, k) since the paths witness that t_i, t_j cannot be separated by removing a single non-terminal.

Hence we need to show how to prove the case when $S' \cap W \neq \emptyset$. In this case, note that due to Lemma 3.1, there is a terminal $t \in T$ and a non-terminal $x \in V(G') \setminus T$ such that the set $S' \cup \{x\}$ is a $(t, T \setminus \{t\})$-separator. So by applying Lemma 3.1 at most $|T| - 1$ times we obtain a set $S^* \subseteq V(G') \setminus T$ such that the set S^* is a mws of (G', T). Moreover, note that $|S^*| \leq |S'| + (|T| - 1)$ because in each step of Lemma 3.1 we add at most 1 additional vertex to separate a terminal. Next, we obtain a new set $\hat{S} := S^* \setminus W$. Now, it is easy to observe that the set $\hat{S} \subseteq V(G') \setminus T$ is a solution of (G', T, k) such that $\hat{S} \cap W = \emptyset$ thus (like in the previous case) we have the property that the set $\hat{S}$ is also a mws of (G, T, k). Hence, (G, T, k) is a YES-instance of mws. ◀