Fixed-parameter tractability of DIRECTED MULTICUT with three terminal pairs parameterized by the size of the cutset: twin-width meets flow-augmentation

Meike Hatzel† Lars Jaffke‡ Paloma T. Lima§ Tomáš Masařík¶
Marcin Pilipczuk† Roohani Sharma∗∗ Manuel Sorge††

Abstract

We show fixed-parameter tractability of the DIRECTED MULTICUT problem with three terminal pairs (with a randomized algorithm). This problem, given a directed graph $G$, pairs of vertices (called terminals) $(s_1,t_1)$, $(s_2,t_2)$, and $(s_3,t_3)$, and an integer $k$, asks to find a set of at most $k$ non-terminal vertices in $G$ that intersect all $s_1t_1$-paths, all $s_2t_2$-paths, and all $s_3t_3$-paths. The parameterized complexity of this case has been open since Chitnis, Cygan, Hajiaghayi, and Marx proved fixed-parameter tractability of the 2-terminal-pairs case at SODA 2012, and Pilipczuk and Wahlström proved the W[1]-hardness of the 4-terminal-pairs case at SODA 2016.

On the technical side, we use two recent developments in parameterized algorithms. Using the technique of directed flow-augmentation [Kim, Kratsch, Pilipczuk, Wahlström, STOC 2022] we cast the problem as a CSP problem with few variables and constraints over a large ordered domain. We observe that this problem can be in turn encoded as an FO model-checking task over a structure consisting of a few 0-1 matrices. We look at this problem through the lenses of twin-width, a recently introduced structural parameter [Bonnet, Kim, Thomassé, Watrigant, FOCS 2020]: By a recent characterization [Bonnet, Giocanti, Ossona de Mendes, Simon, Thomassé, Toruńczyk, STOC 2022] the said FO model-checking task can be done in FPT time if the said matrices have bounded grid rank. To complete the proof, we show an irrelevant vertex rule: If any of the matrices in the said encoding has a large grid minor, a vertex corresponding to the “middle” box in the grid minor can be proclaimed irrelevant — not contained in the sought solution — and thus reduced.

∗The research leading to the results presented in this paper was partially carried out during the Parameterized Algorithms Retreat of the University of Warsaw, PARUW 2022, held in Będlewo in April 2022. This research is a part of projects that have received funding from the European Research Council (ERC) under the European Union’s Horizon 2020 research and innovation programme Grant Agreement 714704 (TM, MP) and 648527 (MH), from the Alexander von Humboldt Foundation (MS), from the Research Council of Norway (LJ), and by a fellowship within the IFI programme of the German Academic Exchange Service (DAAD) (MH).

†National Institute of Informatics, Tokyo, Japan. meikehatzel@nii.ac.jp
‡University of Bergen, Norway. lars.jaffke@uib.no
§IT University of Copenhagen, Denmark. psalt@itu.dk
¶University of Warsaw, Poland. masarik@mimuw.edu.pl
‖University of Warsaw, Poland. m.pilipczuk@mimuw.edu.pl
∗∗Max Planck Institute for Informatics, Saarland Informatics Campus, Saarbrücken, Germany. rsharma@mpi-inf.mpg.de
††TU Wien, Austria. manuel.sorge@ac.tuwien.ac.at
Contents

1 Introduction 1

2 Preliminaries 5
   2.1 Twin-width ................................................. 6
   2.2 (Permutation) CSP .............................................. 6
   2.3 Flow-augmentation ............................................. 7

3 Permutation CSP with bounded twin-width 8

4 Three-terminal-pair DIRECTED MULTICUT is fixed-parameter tractable 10

5 Irrelevant vertex rule—Proof of Lemma 4.8 16

6 Shadow removal 18

7 Harvesting soybeans 22
   7.1 Back to the edge-deletion regime ................................ 22
   7.2 Initial setup ....................................................... 24
   7.3 Initial steps ......................................................... 25
   7.4 Base case ............................................................ 25
   7.5 Small \( \ell \) case ..................................................... 26
   7.6 Large \( \ell \) case ..................................................... 26

8 Two-terminal-pair WEIGHTED DIRECTED MULTICUT is \( W[1] \)-hard 27
   A Hardness of arbitrary CSPs with permutation constraints 33
   B Twin-width to grid-rank 34
1 Introduction

Parameterized complexity studies the existence of fixed-parameter algorithms: algorithms with running time bound by \( f(k) \cdot n^c \), where \( n \) is the size of the input, \( c \) is an arbitrary constant, \( f \) is an arbitrary computable function, and \( k \) is the parameter, which is a selected secondary measure of the input that is intended to reflect the hardness of the instance.

Graph separation problems yield a class of combinatorial problems where the goal is to find a small vertex or edge set in the given graph that satisfies some separation requirements. For example, Multiway Cut equips the input graph \( G \) with a set \( T \subseteq V(G) \) of terminals and asks to cut all paths between any two distinct terminals, whereas Subset Feedback Vertex Set equips the input graph \( G \) with a set \( R \subseteq V(G) \) of red vertices and asks to cut all cycles that contain at least one red vertex. The study of graph separation problems, with the cardinality of the sought cut as the natural choice of the parameter, has been one of the more vivid areas of parameterized complexity in the recent 15 years. A number of interesting algorithmic techniques emerged: important separators [21, 5], applications of matroid techniques [18, 19], shadow removal [9, 24], randomized contractions [6, 11], LP-guided branching [12, 13, 14], and treewidth reduction [23], among others.

The progress somewhat stalled around 5 years ago in the following state: we understood the complexity of the main bulk of graph separation problems in undirected graphs, mostly thanks to the wide variety of algorithmic techniques therein. However, in directed graphs, a number of questions remained widely open.

The MultiCut problem is, given a graph \( G \) and a family \( T \) of pairs of vertices (called terminals, to delete a minimum number of non-terminal vertices so that for every terminal pair \((s, t) \in T\), there is no path from \( s \) to \( t \) in the remaining graph. The parameterized complexity of this problem in undirected graphs, after being a long standing open problem for a while, has been resolved around 2010 independently by two groups of researchers [24, 4]. In directed graphs, the problem in full generality was quickly observed to be W[1]-hard [24]. However, some restrictions turned out to be tractable: the case of directed acyclic graphs [17], Directed Multiway Cut, where we are given just a set of terminals and we ask to cut all paths between every pair of distinct terminals, or Directed Multicut with two terminal pairs [9]. Observe that the one-terminal-pair case is just the classic Minimum Cut problem. In 2015, Pilipczuk and Wahlström [25] provided a hardness reduction for the four-terminal-pairs case, leaving the three-terminal-pairs case open until now.

Directed Multicut with three terminal pairs was by far not the only open problem left in the parameterized complexity of directed graph separation problems. Other open problems included the notoriously difficult Chain SAT problem [7] and most of the problems in the weighted setting. Here, the deletable objects (edges or vertices) have integer weights, and the question is to find a solution of cardinality at most \( k \) and minimum total weight (where \( k \) is the given parameter). One of the reasons for such a state of affairs was a lack of algorithmic techniques in directed graphs: Among all the aforementioned tools in undirected graphs, only important separators and shadow removal generalize to directed graphs [9, 8].

Very recently, at STOC 2022, a new algorithmic technique for cut problems in directed graphs has been presented by Kim, Kratsch, Pilipczuk, and Wahlström [15], namely flow-augmentation. This new technique led to fixed-parameter algorithms for Chain SAT and numerous weighted versions of graph separation problems. In this work, we use it to answer the question of the parameterized complexity of Directed Multicut with three terminal pairs positively.

**Theorem 1.1.** Directed Multicut with three terminal pairs is fixed-parameter tractable when parameterized by the size of the cutset (with a randomized algorithm).

Flow-augmentation alone is by far not enough to show Theorem 1.1. In the basic usage, the main tool of directed flow-augmentation [15] can be stated as follows:
Theorem 1.2. There exists a polynomial-time randomized algorithm that, given a directed graph \( G \), two distinguished vertices \( s, t \in V(G) \), and an integer \( k \), outputs a set \( A \subseteq V(G) \times V(G) \) (called augmentation edges) such that for every minimal edge \( st \)-cut \( Z \) of cardinality at most \( k \), with probability \( 2^{-\Omega(k^4 \log k)} \) the cut \( Z \) becomes a minimum edge \( st \)-cut in \( G + A \).

Here, \( G + A \) is the graph obtained from \( G \) by adding the arcs in \( A \). That is, with good probability, the added arcs \( A \) not only do not break the edge cut \( Z \) (that is, connect the \( s \)-side of the cut to the \( t \)-side), but also increase the connectivity of the graph so that \( Z \) becomes a minimum-cardinality cut. We remark that all randomization in Theorem 1.1 comes from Theorem 1.2. That is, if Theorem 1.2 were deterministic, so would be the algorithm of Theorem 1.1.

The following point of view on Theorem 1.2 turns out to be particularly useful. Let \( G \) be a directed graph, \( s, t \in V(G) \), and let \( k \) be the size of a minimum edge \( st \)-cut. How does the space of all minimum edge \( st \)-cuts look like? Let \( P \) be any maximum \( st \)-flow, seen as a collection of \( k \) edge-disjoint paths from \( s \) to \( t \). Any minimum edge \( st \)-cut contains exactly one edge from each path in \( P \). Furthermore, for every \( P_1, P_2 \in P \) and every \( u_1 \in V(P_1), u_2 \in V(P_2) \), if \( G \) contains a path \( Q \) from \( u_1 \) to \( u_2 \) that does not contain any edge of \( P \), then any minimum edge \( st \)-cut cannot contain an edge of \( P_1 \) after \( u_1 \) and an edge of \( P_2 \) before \( u_2 \) at the same time. This motivates the following CSP formulation. Every path \( P \in P \) becomes a variable \( x(P) \) with domain \( E(P) \), ordered naturally along \( P \). For every tuple \( (P_1, P_2, u_1, u_2) \) as above (i.e., \( P_1, P_2 \in P, u_1 \in V(P_1), u_2 \in V(P_2) \), \( G \) contains a path from \( u_1 \) to \( u_2 \) that does not contain any edge of \( P \)), we introduce a constraint \( (x(P_1) \leq u_1) \lor (x(P_2) \geq u_2) \), where the inequalities have the natural meaning of being before/after the corresponding vertex along the corresponding path. It is relatively easy to see that the space of all feasible solutions to such a CSP instance is exactly the space of all minimum edge \( st \)-cuts in \( G \). In this light, Theorem 1.2 can be understood as follows: we can subsample the space of all minimal edge \( st \)-cuts in \( G \) of cardinality at most \( k \), so that every cut is sampled with good probability (i.e., \( 2^{-\Omega(k^4 \log k)} \)) and the subsampled set can be described by the aforementioned CSP instance.

A meticulous reader may observe that flow-augmentation speaks about edge cuts while Directed Multicut asks for a vertex cut. However, in directed graphs there are standard reductions between these two variants of the problem. Thus, the above framework of a CSP formulation can be easily adapted to minimal vertex \( st \)-cuts of cardinality at most \( k \) (where \( s \) and \( t \) are undeletable).

In the context of Directed Multicut with three terminal pairs, we can use flow-augmentation as follows. Let \( S \) be an inclusion-wise minimal solution to an input instance \( (G, k, (s_i, t_i))_{i=1,2,3} \). Clearly, \( S = S_1 \cup S_2 \cup S_3 \) where \( S_i \) is a minimal \( s_i t_i \)-cut. Hence, we can apply flow-augmentation separately to \( (G, k, (s_i, t_i)) \) for \( i = 1, 2, 3 \), obtaining a set of augmentation edges \( A_i \). With good probability, for every \( i = 1, 2, 3 \) the set \( S_i \) becomes a minimum vertex \( st \)-cut in \( G + A_i \). Let \( P_i \) be a maximum (vertex-capacitated) \( s_i t_i \)-flow in \( G + A_i \) and consider the aforementioned CSP formulation with variables \( \{x_i(P) \mid P \in P_i\} \) and sets of constraints \( C_i \).

The crux of the difficulty of Directed Multicut lies in the fact that the sets \( S_i \) may not be pairwise disjoint; in a sense, we can save on reusing some vertices in \( S \) to separate multiple terminal pairs. In the CSP regime, it means that for some \( i, j \in \{1, 2, 3\} \) and \( P \in P_i \) and \( Q \in P_j \), the variables \( x_i(P) \) and \( x_j(Q) \) describe the same vertex of \( S \). Note that there is only \( 2^{\Omega(k \log k)} \) options of which pairs of variables describe the same vertex; we can exhaustively guess the set \( K \) of all pairs \( (i, j, P, Q) \) as above. Every \( (i, j, P, Q) \in K \) induces a constraint that \( x_i(P) \) and \( x_j(Q) \) is the same vertex; in the CSP language, this is a permutation constraint, denoted henceforth \( C(i, j, P, Q) \), between a subset of the domain of \( x_i(P) \) and a subset of the domain of \( x_j(Q) \).

It is important to observe that — assuming the flow-augmentation steps were successful and in the branching step we made the correct choice of which variables describe the same vertex — the final CSP instance is an equivalent reformulation of the original Directed Multicut instance. That is, every solution to the obtained CSP instance gives a set of non-terminal vertices
that cuts all paths from \( s_i \) to \( t_i \) for \( i = 1, 2, 3 \). The coincidences guessed in the branching step determine the cardinality of the obtained cut, and we can terminate all branches that lead to cuts larger than \( k \). Furthermore, every inclusion-wise minimal solution to the original \textsc{Directed Multicut} instance that is compliant with the flow-augmentation and branching steps yields a feasible solution to the obtained CSP instance. Thus, it “only” remains to solve the obtained CSP instance.

To this end, we need to understand how complex the permutation constraints \( C(i, j, P, Q) \) can be. Note that the number of variables is small — bounded by \( 3k \) — but the domains can be as large as \(|V(G)|\). A reader experienced in \( W[1]\)-hardness reductions may notice at this point that the complexity of the permutation constraints is crucial: If one allows arbitrary permutation constraints (and constraints of the form \((x \leq a) \lor (y \geq b)\) for constants \(a, b\) and variables \(x, y\), as in the encoding of the space of all minimum cuts), one can easily provide a \( W[1]\)-hardness reduction for the parameterization by the number of variables via the edge-choice gadgets (cf. \cite{10} and Appendix A).

Here a recent major milestone in parameterized complexity comes into play: twin-width. Introduced by Bonnet, Kim, Thomassé, and Watrigant in 2020 \cite{3}, this structural complexity measure of graphs and, more generally, binary structures has turned out to explain and provide a number of fixed-parameter tractability results. Most importantly, a recent work from STOC 2022 \cite{2} provides a fixed-parameter algorithm for FO model checking on ordered structures of bounded twin-width and provides a neat characterization under which conditions a 0-1 matrix gives bounded twin-width in the encoding.

In our case, the crucial notion is the one of a grid minor of a matrix.\(^1\) Let \( A \) be a 0-1 matrix of dimension \( n \times m \). An \( \ell \times \ell \) grid minor consists of two sequences of indices \( 0 = i_0 < i_1 < i_2 < \ldots < i_\ell = n \) and \( 0 = j_0 < j_1 < j_2 < \ldots j_\ell = m \) such that for every \( 1 \leq \alpha, \beta \leq \ell \), there is at least one cell \( A[i_\alpha, j_\beta] \) with value 1 for some indices \( i_\alpha, j_\beta \) with \( i_{\alpha-1} < i_\alpha \leq i_{\alpha}\) and \( j_{\beta-1} < j_\beta \leq j_{\beta}\). The value \( \ell \) is often referred to as the size of the grid minor. One can deduce from the results of Bonnet et al. \cite{2} the following statement: Our CSP formulation is FPT when parameterized by the number of variables, the number of permutation constraints, and the maximum size of a grid minor of the matrices of the permutation constraints. Hence, our task boils down to providing a bound on the size of a grid minor in our permutation constraints \( C(i, j, P, Q) \).\(^2\)

To this end and to complete our proof, we prove the following irrelevant-vertex rule: There exists an integer \( \ell \) depending only on \( k \) such that if some constraint \( C(i, j, P, Q) \) has grid minor of size at least \( \ell \), then a vertex corresponding to a 1 in the “middle cell” of the grid minor (i.e., \( A[i_\alpha, j_\beta] \) for \( i_\alpha < i_\beta \)) is irrelevant, that is, any so-called shadowless solution to the input \textsc{Directed Multicut} instance does not contain the said vertex. Hence, such a vertex can be reduced in a standard manner (and the analysis of the CSP formulation restarted).

The notion of a shadowless solution comes from the technique of shadow removal, pivotal for the fixed-parameter algorithms for \textsc{Directed Multiway Cut} \cite{9} and \textsc{Directed Subset Feedback Vertex Set} \cite{8}. In the context of \textsc{Directed Multicut}, a solution \( Z \) is shadowless if for every non-terminal vertex \( v \notin Z \), the graph \( G - Z \) features a path from \( v \) to one of the terminals \( t_i \) and a path from one of the terminals \( s_j \) to \( v \) (note that necessarily the \( t \)-terminal and the \( s \)-terminal have distinct indices). In short, the shadow-removal technique \cite{9, 8} allows us to focus on the following task: Given a \textsc{Directed Multicut} instance \((G, k, (s_i, t_i)_{i=1,2,3})\), find any solution if there exists a shadowless solution (i.e., the algorithm is allowed to fail if there is a solution, but not a shadowless one).

---

\(^1\)Note that this is an entirely different concept to a grid minor of a graph.

\(^2\)The reader experienced in \( W[1]\)-hardness reductions may recall at this point the reduction showing \( W[1]\)-hardness parameterized by the number of variables for CSPs with permutation constraints: The permutations used therein for the edge-choice gadgets are of the form \( \pi : \{0, 1, \ldots, n^2 - 1\} \rightarrow \{0, 1, \ldots, n^2 - 1\} \), with \( \pi(xn+y) = yn+x \) for \( 0 \leq x, y < n \). These are exactly the permutations with largest possible grid minors of their associated permutation matrices.
This last part of the proof — the irrelevant-vertex rule — is the only part of the proof that crucially relies on the fact that we are dealing with only three terminal pairs. In fact, it is inspired by the reduction for four terminal pairs [25] and our study why this reduction fails for three terminal pairs.

The irrelevant-vertex rule also requires us to look into the details of flow-augmentation (Theorem 1.2) and extract some extra properties of this tool. In short, we need to capture the following intuition: A sequence of deletable edges along the same flow path \( P \in \mathcal{P} \) in the maximum flow \( P \) in \( G + A \) is in some sense sequentially positioned in the graph, so one can usually reach any later edge from an earlier one. This is not strictly true as stated above, but we prove a variant of this statement in Section 7.

One can ask if the proof of the irrelevant-vertex rule crucially needs the assumption of the solution being shadowless. In particular, the usage of shadow removal makes our algorithm inherently unweighted (the shadow-removal step involves a greedy argument that completely breaks down in the presence of weights). We complement our main result by proving (Section 8) that \textsc{Weighted Directed Multicut} is W[1]-hard even with two terminal pairs, so the shadow-removal step seems necessary.

**Theorem 1.3.** \textsc{Weighted Directed Multicut}, parameterized by the cardinality of the cut-set, is W[1]-hard even with two terminal pairs.

We remark that the one-terminal-pair case of \textsc{Weighted Directed Multicut}, or Bi-objective \( st \)-cut, is proved to be FPT in [15] as one of the basic exemplary usages of flow-augmentation.

Since we rely on the whole meta-algorithmic toolbox of twin-width of [3], we cannot state an explicit dependency on the parameter in the running time bound of our algorithm. Relatedly, we would like to remark that the concept of encoding an instance at hand into a CSP instance with a number of variables that is bounded by a function of the parameter, but over large ordered domains, appeared also recently in the FPT algorithm for \textsc{Optimal Discretization} [16]. The encoding there also uses an unbounded number of constraints of the form \((x \leq a) \lor (y \geq b)\) and a bounded-in-parameter number of permutation constraints. However, the main effort in the proof in [16] lies in showing that the used permutation constraints have very simple structure (they are called in [16] segment reversions); in particular, one can observe that their permutation matrices do not contain a grid minor of size 3. As a result, in [16] the authors are able to design an explicit FPT algorithm for the obtained CSP instance (with an explicit single-exponential running time bound). Although using twin-width meta-algorithms prevents us from stating an explicit running time bound, it allows to claim fixed-parameter tractability of a much wider range of CSP instances: Permutations of bounded grid minor in their permutation matrices is a much wider class than the aforementioned segment reversions. We believe the presented framework of casting a problem into a small number of variables with unbounded ordered domains, bound by permutation constraints, and using twin-width toolbox to solve it, has a wider future potential in parameterized complexity.

**Organization.** After brief preliminaries, where in particular we state the shadow-removal tool and the extended version of flow-augmentation, we proceed to the main proof (of Theorem 1.1). Section 3 introduces the twin-width toolbox and shows fixed-parameter tractability of CSP instances with permutations only containing bounded grid minors. Section 4 contains the main proof, deferring the proof of the irrelevant-vertex rule to Section 5. Section 6 contains a (standard) proof of the used shadow-removal statement, while Section 7 contains a proof of the used extension of flow-augmentation. Finally, the proof of Theorem 1.3 can be found in Section 8.
**Acknowledgements.** The research leading to the results presented in this paper was partially carried out during the Parameterized Algorithms Retreat of the University of Warsaw, PARUW 2022, held in Będlewo in April 2022. We acknowledge insightful discussions with the twin-width experts at Będlewo: Édouard Bonnet, Jakub Gajarský, and Michał Pilipczuk, as well as later discussions with Szymon Toruńczyk.

## 2 Preliminaries

Let $G$ be a directed graph. We use edge and arc interchangeably for the directed edges of $G$. For two vertices $u,v \in V(G)$, we say that $u$ reaches $v$ or $v$ is reachable from $u$ if there exists a directed path from $u$ to $v$ in $G$. A path starting in a vertex $u$ and ending in a vertex $v$ is also called a $uv$-path. Let $e$ be a (directed) edge. Then $s(e)$ ($t(e)$) is start (target) of $e$. Analogously, we use $s(P)$ (resp. $t(P)$) for first (resp. last) vertex of a path $P$. For $s,t \in V(G)$, a set $S \subseteq E(G)$ is an $st$-cut if $t$ is not reachable from $s$ in $G - S$ and, similarly, a set $S \subseteq V(G) \setminus \{s,t\}$ is an $st$-separator if $t$ is not reachable from $s$ in $G - S$. In the latter, by $G - S$ we denote the subgraph of $G$ induced by $V(G) \setminus S$. Throughout the paper we use $[k]$ as shorthand for the integer set $\{1,2,\ldots, k\}$.

In this work, an instance of **DIRECTED MULTICUT** ($3$-DMC for short) is a tuple $(G,k,(s_i,t_i)_{i=1,2,3},V^\infty)$ consisting of a directed graph $G$, six distinguished vertices $s_1,s_2,s_3,t_1,t_2,t_3 \in V(G)$, called terminals, an integer $k$, and a vertex subset $V^\infty \subseteq V(G)$, called undeletable vertices. We sometimes denote the set of terminals as $T = \{s_1,s_2,s_3,t_1,t_2,t_3\}$. We require $T \subseteq V^\infty$. A solution is a set $S$ of non-terminal vertices of $G$ such that $S \cap V^\infty = \emptyset$ and for every $i = 1,2,3$ the vertex $t_i$ is not reachable from the vertex $s_i$ in $G - S$. **DIRECTED MULTICUT** asks for a solution of cardinality at most $k$.

For an instance $(G,k,(s_i,t_i)_{i=1,2,3},V^\infty)$ and a non-terminal vertex $v$, by bypassing $v$ we mean the following operation. First, for every edge $(u,v) \in E(G)$ and every edge $(v,w) \in E(G)$, we add an edge $(u,w)$ if it is not already present in the graph. Finally, we delete the vertex $v$. For a set $X$ of nonterminal vertices, by bypassing $X$ we mean bypassing vertices of $X$ in arbitrary order; note that the result does not depend on the order. The following lemma is immediate and shows that bypassing is a good way to reduce vertices that are provably not in the sought solution.

**Lemma 2.1** (Chitnis et al. [8], Lemma 3.11). Let $G$ be a directed graph and let $C \subseteq V(G)$. Let $G'$ be obtained from $G$ after bypassing $C$ and let $S \subseteq V(G) \setminus C$. For any $a,b \in V(G) \setminus (C \cup S)$, $G - S$ has an $ab$-path if and only if $G' - S$ has an $ab$-path.

**Shadows and shadow removal.** Given an instance $(G,k,(s_i,t_i)_{i=1,2,3},V^\infty)$ of $3$-DMC and a set $X \subseteq V(G) \setminus V^\infty$, a vertex $v \in V(G) \setminus (X \cup V^\infty)$ is in the forward shadow of $X$ if $v$ is reachable from neither $s_1, s_2$, nor $s_3$ in $G - X$. Symmetrically, $v$ is in the reverse shadow of $X$ if neither $t_1, t_2$, nor $t_3$ is reachable from $v$ in $G - X$. The set of vertices in the forward shadow of $X$ in $G$, is denoted by $f_G(X)$ and the set of vertices in the reverse shadow of $X$ in $G$, is denoted by $r_G(X)$. The vertex $v$ is in the shadow of $X$ if it is in the forward shadow or the reverse shadow of $X$. A set $X$ is **shadowless** if no vertex is in its shadow, that is, if $f_G(X) \cup r_G(X) = \emptyset$.

The following statement encapsulates the shadow removal technique in the context of **DIRECTED MULTICUT**. Though it follows directly from [9, 8], we provide a formal proof in Section 6 for the sake of completeness.

**Theorem 2.2.** Given an instance $(G,k,(s_i,t_i)_{i=1,2,3},V^\infty)$ of $3$-DMC, there is an algorithm that runs in time $2^{O(k^2 \log k)} n^{O(1)}$, and outputs a family $Z$ of subsets of $V(G) \setminus V^\infty$ such that $|Z| = 2^{O(k^2 \log k)} \log^2 n$ and, if the input instance is a Yes-instance, then there exists $Z \in Z$ such that $(G',k,(s_i,t_i)_{i=1,2,3},V^\infty \setminus Z)$ is a Yes-instance that admits a shadowless solution of cardinality at most $k$, where $G'$ is the result of bypassing $Z$ in $G$. 5
2.1 Twin-width

Next, we state the definition of twin-width as introduced by Bonnet et al. [3]. They make use of the concept of a 
trigraph $G$, which consists of a vertex set $V(G)$, and two disjoint edge sets, one containing black edges $E(G)$ and the other containing red edges $R(G)$. In particular, every graph $G$ is a trigraph with only black edges and $R(G)$ being empty. Let $G$ be a trigraph. We say that we contract two vertices $u, v \in V(G)$ if we merge them into a single vertex $w$, and then possibly color the edges incident to the new vertex $w$. Every existing edge $uw$ remains black if and only if $uz$ and $vz$ were previously black edges. All other edges incident to $w$ are colored in red. A contraction sequence of an $n$-vertex (tri)graph $G$ is a sequence of trigraphs $G = G_0, \ldots, G_1 = K_1$ such that $G_i$ is obtained from $G_{i+1}$ by contracting two vertices. A contraction sequence is called a $d$-sequence if all trigraphs in it have red degree at most $d$. The twin-width of $G$, denoted by $\text{tww}(G)$, is the minimum integer $d$ such that $G$ admits a $d$-sequence.

Now we turn to matrices. Here, our main proxy to the twin-width are so-called rank-$k$ divisions and the grid rank of the matrix which are closely related to twin-width graph parameter, see Bonnet et al. [2]. Let $M$ be a 0-1 matrix. A division $D$ of $M$ is a pair $(D^R, D^C)$, where $D^R$ and $D^C$ are partitions of the rows and columns into intervals of consecutive rows and intervals of consecutive columns, respectively. A $k$-division is a division with $|D^R| = |D^C| = k$. If $(D^R, D^C)$ is a division, $D^R = (R_1, R_2, \ldots)$ and $D^C = (C_1, C_2, \ldots)$, then for each pair $R_i, C_j$, the (contiguous) submatrix of $R_i \cap C_j$ is called the $(i,j)$-cell of $D$. A rank-$k$ division of $M$ is a $k$-division $D$ of $M$ such that each cell of $D$ contains at least $k$ distinct rows and at least $k$ distinct columns, or, equivalently, has combinatorial rank at least $k$. The maximum integer $k$ such that $M$ admits a rank-$k$ division is called the grid rank of $M$, denoted by $\text{gr}(M)$. A $k$-grid minor of $M$ is a $k$-division $D$ of $M$ such that each cell of $D$ contains at least one 1; note that any rank-$k$ division of $M$ for $k \geq 2$ is necessarily a $k$-grid minor, too.

Let $G$ be a graph and $\prec$ be a linear order on $V(G)$. We denote by $\text{Adj}_\prec(G)$ the adjacency matrix of $G$ where rows and columns are ordered according to $\prec$. We use the following.

**Theorem 2.3** (Bonnet et al. [2]). There is a computable function $f : \mathbb{N} \to \mathbb{N}$ such that the following holds. Let $G$ be a graph.

- For any total order $\prec$ of $V(G)$, if $\text{gr}(\text{Adj}_\prec(G)) \leq k$, then $\text{tww}(G) \leq f(k)$.
- If $\text{tww}(G) \leq k$, then there is a total order $\prec$ of $V(G)$ such that $\text{gr}(\text{Adj}_\prec(G)) \leq f(k)$.

**Lemma 2.4** (Bonnet et al. [1]). There is a computable function $f : \mathbb{N} \to \mathbb{N}$ such that the following holds. Let $V$ be a set of vertices, let $\prec$ be a linear order on $V$, and let $G_1$ and $G_2$ be two graphs on the vertex set $V$. If $\text{gr}(\text{Adj}_\prec(G_i)) \leq k$ for all $i \in [2]$, then $\text{gr}(\text{Adj}_\prec(G_1 \cup G_2)) \leq f(k)$.

The following theorem can be deduced from the arguments of [3], but is not stated there explicitly; for completeness, we provide a proof in Appendix B.

**Theorem 2.5.** There is a computable function $f$ with $f(k) = 2^{O(k \log k)}$ such that the following holds. There is an algorithm that, given a 0-1 matrix $A$, in $f(k)n^{O(1)}$ time either

- finds a $k$-grid minor in $A$, or
- certifies that $\text{gr}(A) \leq f(k)$.

2.2 (Permutation) CSP

An instance of a constraint satisfaction problem (CSP) is a triple $(X,D,C)$, where $X = \{x_1, \ldots, x_n\}$ is a set of variables, $D = \{D_1, \ldots, D_n\}$ a set of domains, and each $C \in C$ a constraint. A constraint $C$ is an $a_C$-tuple $(x_{a_1}, \ldots, x_{a_C})$ of variables and a relation $R(C) \subseteq D_{a_1} \times \ldots \times D_{a_C}$. A valuation $\alpha$ assigns to each $x_i$ a value $\alpha(x_i) \in D_i$. A constraint $C \in C$ is satisfied by $\alpha$ if $(\alpha(x_{a_1}), \ldots, \alpha(x_{a_C})) \in R(C)$. A valuation $\alpha$ satisfies $(X,D,C)$ if it satisfies all constraints.
Let \((D_1, \leq_1)\) and \((D_2, \leq_2)\) be finite totally ordered sets. We need two types of relations in \(D_1 \times D_2\). A relation \(R \subseteq D_1 \times D_2\) is called downwards-closed if for every \((x_1, x_2) \in R\) and every \(x'_1 \leq_1 x_1\) and \(x'_2 \leq_2 x_2\) it holds that \((x'_1, x'_2) \in R\). Let \(X_1 \subseteq D_1\) and \(X_2 \subseteq D_2\) with \(|X_1| = |X_2|\) and let \(\pi: X_1 \rightarrow X_2\) be a bijection. We refer to the relation \(\{(x, \pi(x)) \mid x \in X_1\}\) as the permutation constraint \(\pi\). We denote by \(\text{Adj}(\pi)\) the 0-1 matrix of dimension \(|D_1| \times |D_2|\) associated with \(\pi\) as follows. The rows and columns one-to-one correspond to \(D_1\) and \(D_2\), respectively, following the orders \(\leq_1\) and \(\leq_2\). The entry associated with \((x, y) \in D_1 \times D_2\) equals 1 if and only if \(x \in D_1\) and \(y = \pi(x)\). We deal with CSPs of the following form.

**Definition 2.6.** A Twin-width-\(w\) Permutation CSP instance consists of variables \(x_1, \ldots, x_k\) with domains \((D_1, \leq_1), \ldots, (D_k, \leq_k)\), where for all \(i \in [k]\), \((D_i, \leq_i)\) is a totally ordered set, and each constraint is, for some \(i \neq j\), either
- a downwards-closed relation \(R \subseteq D_i \times D_j\), or
- a permutation constraint \(\pi: X_i \rightarrow X_j\) where \(X_i \subseteq D_i\) and \(X_j \subseteq D_j\).

Furthermore, for each permutation constraint \(\pi\), we have that \(\text{gr}(\text{Adj}(\pi)) \leq w\).

### 2.3 Flow-augmentation

Because the shadow removal technique is much easier to phrase and use in the vertex-deletion regime, we need to adjust flow-augmentation [15] from edge deletions to vertex deletions. This is pretty straightforward via the standard reductions between edge- and vertex-deletion regimes in directed graphs. More importantly, we need to squeeze an extra connectivity property out of flow-augmentation, which we formalize below as soybeans. We present here only the necessary definitions and the main statement that is used in the algorithm; its proof is deferred to Section 7.

Let \(G\) be a directed graph with two distinguished vertices \(s, t \in V(G)\). The vertices of \(G\) are partitioned into deleteable vertices and undeletable vertices; \(s\) and \(t\) are undeletable. A (vertex-based) st-flow is a collection of st-paths that do not share a deleteable vertex; the number of paths is the value of the flow. We use \(\lambda_G(s, t)\) for the maximum possible value of an st-flow; a flow of value \(\lambda_G(s, t)\) is an st-maxflow. As a convention we say that, if \(G\) contains an st-path consisting of undeletable vertices only, then the flow containing such a path has value \(+\infty\) and \(\lambda_G(s, t) = +\infty\).

A set \(Z\) of deleteable vertices is an st-separation if there is no path from \(s\) to \(t\) in \(G - Z\). An st-separation \(Z\) is minimal if no proper subset of \(Z\) is an st-separation and minimum (or st-mincut) if it has minimum possible cardinality. By Menger’s theorem, if \(\lambda_G(s, t) < +\infty\) then the size of every st-mincut is exactly \(\lambda_G(s, t)\) and there are no st-separations if \(\lambda_G(s, t) = +\infty\). We drop the subscript if the graph \(G\) is clear from the context.

We say that a set of arcs \(A \subseteq V(G) \times V(G)\) is compatible with a minimal st-separation \(Z\) if the following holds: for every \(v \in V(G)\), there is a path from \(s\) to \(v\) in \(G - Z\) if and only if there is a path from \(s\) to \(v\) in \((G + A) - Z\). The pair \((A, P)\) is compatible with \(Z\) if \(A\) is compatible with \(Z\) and \(P\) is an st-maxflow in \(G + A\).

A soybean in \(G\) is an unordered pair of walks that have the same starting vertex and the same ending vertex. We do not require the walks to be disjoint in any sense; in particular, a pair consisting of the same walk twice is always a soybean. Two soybeans \(PQ\) and \(PQ'\) are vertex-disjoint if \((V(P) \cup V(Q)) \cap (V(P') \cup V(Q')) = \emptyset\). For two sets of vertices or edges \(C, D \subseteq V(G) \cup E(G)\), a soybean \(PQ\) is a CD-soybean if one walk of \(PQ\) contains an edge or a vertex of \(C\) and the other walk of \(PQ\) contains an edge or a vertex of \(D\). For a path \(P\) and two disjoint sets \(C, D \subseteq V(P) \cup E(P)\), we say that \(C\) and \(D\) are interlaced on \(P\) if \(|C| = |D| = q\) for some integer \(q\), \(C\) can be enumerated as \(c_1, \ldots, c_q\), \(D\) can be enumerated as \(d_1, \ldots, d_q\), and the order of these vertices and edges along \(P\) is \(c_1, d_1, c_2, d_2, \ldots, c_q, d_q\).

**Theorem 2.7.** There exist computable functions \(c: \mathbb{N} \rightarrow \mathbb{N}\) and \(q: \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}\) such that the following holds.
There exists a polynomial-time randomized algorithm that, given a directed graph $G$ (with possibly some vertices marked as undeletable), vertices $s,t \in V(G)$, and an integer $k$, returns an arc set $A \subseteq V(G) \times V(G)$ and an st-maxflow $\mathcal{P}$ in $G+A$ such that for every minimal st-separator $Z$ of size at most $k$, with probability $2^{-O(k^3 \log k)}$, the tuple $(A, \mathcal{P})$ is compatible with $Z$.

Additionally, the algorithm returns a partition $B$ of the deletable vertices of $\bigcup_{P \in \mathcal{P}} V(P)$ into at most $c(k)$ sets such that for every $P \in \mathcal{P}$, every integer $p \in \mathbb{N}$, every $B \in B$ and every two disjoint sets $C, D$ of size at least $q(k,p)$, consisting of vertices of $B \cap V(P)$ that are interlaced on $P$, the graph $G$ contains a family of $p$ pairwise vertex-disjoint $CD$-soybeans.

Finally, one can take $c$ and $q$ such that $c(k) = 2^{O(k^3 \log k)}$ and $q(k,p) = 2^{O(k^3 \log (kp))}$.

3 Permutation CSP with bounded twin-width

In this section we show the following.

**Theorem 3.1.** Twin-width-$w$ Permutation CSP parameterized by the number of constraints plus $w$ is fixed-parameter tractable.

**Proof.** We transform the given Twin-width-$w$ Permutation CSP instance $\mathcal{I}$ into an ordered, vertex- and edge-colored graph $G$ whose twin-width only depends on $w$ and the number of constraints in $\mathcal{I}$. We then give an FO-formula $\phi$ such that $G \models \phi$ if and only if $\mathcal{I}$ is satisfiable. We can then apply the FO model checking algorithm on (ordered) graphs that runs in FPT time when parameterized by the twin-width of the input graph plus the length of the formula [2, 3].

We denote the variables of $\mathcal{I}$ by $x_1, \ldots, x_k$, by $\mathcal{R}$ the set of downwards-closed constraints of $\mathcal{I}$, and by $\Pi$ the set of permutation constraints of $\mathcal{I}$. For each $i \in [k]$, let $d_i^1, \ldots, d_i^{n_i}$ denote the elements of $D_i$, ordered according to $\leq_i$. For each $i \in [k]$, we introduce into $G$ a set $V_i$ of $n_i$ vertices, colored with color $i$ and denoted (and ordered) as $v_i^1, \ldots, v_i^{n_i}$. We reuse the symbols “$\leq_i$” to denote the ordering of the vertices in $V_i$ in the FO-formula. At the heart of our FO-formula $\phi$ is an existential guess of one vertex per $V_i$, and choosing $v_j^i$ for $j \in [n_i]$ corresponds to setting the variable $x_i$ to $d_j^i$. The vertex colors are $[k]$, and encode membership in the $V_i$’s. We assume we have predicates $\text{col}(y) = i$ for a vertex variable $y$ and $i \in [k]$ that verify whether the vertex assigned to $y$ has color $i$ (which in turn means that $y \in V_i$). The existential guess is:

$$\phi_3 \equiv \exists y_1 \ldots \exists y_k \bigwedge_{i \in [k]} \text{col}(y_i) = i$$

**Encoding the constraints in $(G, \phi)$**. Next we add edges to the graph $G$ and color them using the constraints $\mathcal{R} \cup \Pi$ of $\mathcal{I}$ as colors, thus indicating why an edge was added to $G$. Similarly to above, we assume that we have predicates $\text{col}(e) = C$ where $C \in \mathcal{R} \cup \Pi$, that verify whether the edge assigned to $e$ has color $C$. Let $R \in \mathcal{R}$ be a downwards-closed constraint with $R \subseteq D_i \times D_j$. For each $(d^i_a, d^i_b) \in R$ such that there is no $(a', b') \in [n_i] \times [n_j] \setminus \{(a,b)\}$ with $a' \geq a$, $b' \geq b$, and $(d_i^{a'}, d_i^{b'}) \in R$, we add the edge $v_i^a v_i^b$ colored $R$ to $G$. Note that the edges with color $R$ form a matching, and that they mark the “boundary” of the relation $R$, in the following sense. For each $(d^i_a, d^i_b) \in D_i \times D_j$, we have that $(d^i_a, d^i_b) \in R$ if and only if there is an edge $(v_i^a, v_i^b)$ with $a' \geq_i a$, $b' \geq_j b$, and of color $R$ in $G$. We construct the following part of $\phi$ which checks this condition, and therefore is true if and only if $R$ is satisfied under the value assignment to $x_1, \ldots, x_k$ corresponding to the choice of the vertices $y_1, \ldots, y_k$.

$$\phi_R \equiv \exists e \exists z_i \exists z_j \left( \text{col}(e) = R \land \bigwedge_{h \in \{i,j\}} \text{inc}(e, z_h) \land \text{col}(z_h) = h \land y_h \leq_h z_h \right)$$

Now, let $\pi \in \Pi$ with $\pi: X_i \rightarrow X_j$ for some $X_i \subseteq D_i$ and $X_j \subseteq D_j$ be a permutation constraint. Then, for each $d_i^a \in X_i$, we let $d_i^b = \pi(d_i^a)$, and add the edge $v_i^a v_i^b$ colored $\pi$ to $G$. Again, the
edges with color \( \pi \) form a matching. This finishes the construction of \( G \), and the last building block of \( \phi \) is as follows. It checks that whenever \( y_i \) (resp. \( y_j \)) is incident with some edge colored \( \pi \), that the other endpoint of that edge is chosen to be \( y_j \) (resp. \( y_i \)). This part of the formula evaluates to true if and only if the permutation constraint \( \pi \) is satisfied under the corresponding choice of values for the variables of \( I \).

\[
\phi_\pi \equiv \exists e \left( \text{col}(e) = \pi \land \text{inc}(e, y_i) \land \text{inc}(e, y_j) \right).
\]

We now obtain \( \phi \) as

\[
\phi \equiv \phi_3 \land \bigwedge_{R \in \mathcal{R}} \phi_R \land \bigwedge_{\pi \in \Pi} \phi_\pi.
\]

The correctness of this transformation immediately follows from the description above:

Claim 3.2. \( I \) is satisfiable if and only if \( G \models \phi \).

It remains to show that the twin-width of \( G \) is bounded by a function of \( w \) and the number of constraints in \( I \). To do that, consider the order \( V_1 < V_2 < \ldots < V_k \) of \( V(G) \), where each \( V_i \) is ordered according to \( \leq_i \); denote this order by \( \prec \). We want to show that \( \text{gr} (\text{Adj}_\prec(G)) \leq h(w, |R \cup \Pi|) \), for some computable function \( h \), which implies a desired bound on the twin-width of \( G \) by Theorem 2.3. This can be done by repeated application of Lemma 2.4 over all edge colors of \( G \). All permutation constraints have grid rank at most \( w \) by assumption, so it remains to show that the grid rank of downwards-closed constraints is bounded as well. Before we do so, we observe one minor technical detail.

Claim 3.3. Let \( (V, \prec) \) be an ordered set of vertices and let \( X_i \) and \( X_j \) be disjoint consecutive subsets of \( V \). Let \( G \) be a graph on the vertex set \( V \) that only has edges between \( X_i \) and \( X_j \). For each \( k \geq 1 \), if \( \text{gr} (\text{Adj}_\prec(G)) \geq k + 2 \), then \( \text{gr} (\text{Adj}_\prec(G)[X_i, X_j]) \geq k \).

Proof. Let \( M = \text{Adj}_\prec(G) \), \( M_{i,j} = M[X_i, X_j] \), and \( M_{i,j} = M[X_i, X_j] \). Let \( D = (D^R, D^C) \) be a rank-\((k+2)\) division of \( M \). Since outside of \( M_{i,j} \) and \( M_{j,i} \), \( M \) is all-0, for either \( M_{i,j} \) or \( M_{j,i} \), we may assume (up to renaming \( i \) and \( j \)) that every cell of \( D \) intersects \( M_{i,j} \). This implies that \( k^2 \) cells of \( D \) are entirely contained in \( M_{i,j} \). Since each such cell has combinatorial rank at least \( k + 2 \), this gives a rank-\( k \) division of \( M_{i,j} \).

Claim 3.4. Let \( R \in \mathcal{R} \) be a downwards-closed constraint, let \( E_R \subseteq E(G) \) be the set of edges colored \( R \) in \( G \), and let \( G_R = (V(G), E_R) \). Then, \( \text{gr} (\text{Adj}_\prec(G_R)) \leq 3 \).

Proof. Let \( i, j \in [k] \) be such that \( R \subseteq D_i \times D_j \), and let \( M_{i,j} = \text{Adj}_\prec(G_R)[V_i, V_j] \). Suppose for a contradiction that there is a rank-2 division of \( M_{i,j} \) with row intervals \( (R_1, R_2) \) and column intervals \( (C_1, C_2) \). Since each cell of this division has combinatorial rank at least \( 2 \), we know that each such cell contains at least one 1. Let \( e_1 = u_1v_1 \) be the edge corresponding to the 1 in \( R_1 \cap C_1 \), and let \( e_2 = u_2v_2 \) be the edge corresponding to the 1 in \( R_2 \cap C_2 \), where \( u_1, u_2 \in V_i \) and \( v_1, v_2 \in V_j \). But then, \( u_1 <_i u_2 \) and \( v_1 <_j v_2 \), which means that the above construction would not have added the edge \( e_1 \) with color \( R \). This implies that \( \text{gr} (M_{i,j}) \leq 1 \), so by Claim 3.3, \( \text{gr} (\text{Adj}_\prec(G_R)) \leq 3 \).

Claim 3.5. There is a computable function \( h : \mathbb{N} \times \mathbb{N} \to \mathbb{N} \), such that \( \text{tww}(G) \leq h(w, |R \cup \Pi|) \).

Proof. For each \( C \in \mathcal{R} \cup \Pi \), let \( E_C \subseteq E(G) \) denote the set of edges colored \( C \), and let \( G_C = (V(G), E_C) \). Then, \( \mathcal{E} = \{ E_C \mid C \in \mathcal{R} \cup \Pi \} \) is a partition of \( E(G) \). Recall that \( \prec \) is a linear order on \( V(G) \) which lets \( V_1 < V_2 < \ldots < V_k \), and for each \( i \in [k] \), coincides with \( \leq_i \) on \( V_i \). By the assumption of the theorem, and Claim 3.4, we have that \( \text{gr} (\text{Adj}_\prec(G)) \leq \max \{ 3, w \} \) for all \( C \). Since \( \mathcal{E} \) is a partition of the edge set of \( G \), we can repeatedly apply Lemma 2.4 to conclude that \( \text{gr} (\text{Adj}_\prec(G)) \leq h'(w, |R \cup \Pi|) \) for some computable function \( h' \). Theorem 2.3 in turn implies that \( \text{tww}(G) \leq h(w, |R \cup \Pi|) \) for some computable function \( h \).
We can now run the FO model checking algorithm \[2, 3\] on \((G, \phi)\) and return the same answer. Note that by \[2\], we can compute a contraction sequence of \(G\) whose twin-width is bounded by a computable function of \(\text{tww}(G)\) in FPT time parameterized by \(\text{tww}(G)\), using the ordering \(<\). Correctness follows from Claim 3.2, and by Claim 3.5 the twin-width of \(G\) only depends on \(w\) and the number of constraints \(s\) in \(I\). It is clear that the length of \(\phi\) can be upper bounded by a function of \(k\) and \(s\), and since we can assume that \(k \leq s\), the length of \(\phi\) can be upper bounded by a function of \(s\) alone. The algorithm of \[2, 3\] is fixed-parameter tractable in \(\text{tww}(G) + |\phi|\), which in our application translates to an FPT-algorithm for Twin-width-\(w\) Permutation CSP parameterized by \(w + s\), as desired. \(\square\)

We would like to remark that Theorem 3.1 generalizes the fixed-parameter tractability result of another type of CSP, called Forest CSP [16]. Moreover, if we drop the twin-width of the permutation constraints as part of the parameter, then the resulting Permutation CSP problem parameterized by the number of constraints is \(W[1]\)-hard (see Appendix A).

4 Three-terminal-pair DIRECTED MULTICUT is fixed-parameter tractable

This section is dedicated to proving the main theorem of this paper:

**Theorem 1.1.** DIRECTED MULTICUT with three terminal pairs is fixed-parameter tractable when parameterized by the size of the cutset (with a randomized algorithm).

To prove Theorem 1.1, we show how to reduce 3-DMC to Permutation CSP such that the twin-width of each constraint is bounded by some function of the desired separator size \(k\). One main ingredient in this reduction is the flow-augmentation technique explained in Section 2.3. The outline is as follows. We first perform flow-augmentation for each of the terminal pairs, giving us an augmented graph and a flow of value \(k\) for each terminal pair. If there is a solution \(S\), then with large-enough probability \(S\) is preserved as a separator after the augmentation steps. The solution thus corresponds to a selection of vertices, one for each of the obtained flow paths. To obtain a reduction to Permutation CSP, the idea is then to introduce one variable \(x\) for each flow path \(P\) where the domain of \(x\) is the set of vertices on \(P\). A set of straightforward constraints ensures that the vertices selected by the variables form a separator for each terminal pair. One crux with this approach is how to ensure that the variables introduced for flow paths between different terminal pairs select vertices in a consistent way. We note that, after trying all possibilities of the possible overlaps of selected vertices, the consistency requirement can be modelled as a permutation constraint, we also call these consistency constraints. In this way, we obtain an instance of Permutation CSP with \(O(k^2)\) constraints. However, as mentioned in Section 3, Permutation CSP in general is \(W[1]\)-hard with respect to the number of constraints. Thus, we need more work to obtain Theorem 1.1. We show how to ensure that the obtained constraints are simple, in the sense that they have low twin-width, crucially leveraging our improved version of flow-augmentation from Section 2.3. We then apply the algorithm from Section 3 for solving Permutation CSP instances of low twin-width.

For use in the remainder of the section, fix an instance \((G, (s_i, t_i)_{i \in [3]}, k)\) of 3-DMC. We show how to solve this instance in fixed-parameter time with large-enough probability. The algorithm is partitioned into four main steps; they are reflected in the structure of the remainder of this section, also see Figure 1 for an illustration. The first step is the shadow-removal technique, which ensures in fixed-parameter time that each vertex is reachable from some terminal and reaches some terminal. We crucially use this property when bounding the twin-width later on. The second step is using flow-augmentation and reducing to Permutation CSP. The third
The first step in the algorithm is to remove vertices from the graph in Shadow removal. To ensure that if there is a solution, then there is also one without a shadow. Recall the definition of being shadowless from Section 2 and recall Theorem 2.2 which we restate for order to ensure that if there is a solution, then there is also one without a shadow. Recall results from the other sections are used.

**Theorem 2.2.** Given an instance \((G,k,(s_i,t_i)_{i\in[3]},V^\infty)\) of 3-DMC, there is an algorithm that runs in time \(2^{O(k^2 \log k)} n^{O(1)}\), and outputs a family \(Z\) of subsets of \(V(G) \setminus V^\infty\) such that \(|Z| = 2^{O(k^2 \log k)} \log^2 n\) and, if the input instance is a \(\mathsf{YES}\)-instance, then there exists \(Z \in Z\) such that \((G',k,(s_i,t_i)_{i\in[3]},V^\infty \setminus Z)\) is a \(\mathsf{YES}\)-instance that admits a shadowless solution of cardinality at most \(k\), where \(G'\) is the result of bypassing \(Z\) in \(G\).

We apply the algorithm of Theorem 2.2, yielding the family \(Z\). We then iterate over all instances of 3-DMC resulting from bypassing a set \(Z \in Z\) in the input graph \(G\). For each such constructed instance we continue with the remainder of the algorithm as described below. For simplicity, we call the instance of the current iteration \((G,k,(s_i,t_i)_{i\in[3]},V^\infty)\). Note that, from now on, it is enough to find a shadowless solution. This is not immediately relevant, but we crucially use this property when bounding the twin-width later on.

**Flow-augmentation and reduction to Permutation CSP.** We continue with the reduction to Permutation CSP, however, without at first bounding the twin-width of all constraints. (Recall that the definition of Permutation CSP can be found in Section 2.2 as Definition 2.6.)

Recall that \((G,k,(s_i,t_i)_{i\in[3]},V^\infty)\) is the instance of 3-DMC that we are working on. The reduction to Permutation CSP works as follows. For each terminal pair \((s_i,t_i), i \in [3]\), run the algorithm from Theorem 2.7 with input \(s = s_i, t = t_i\), the graph \(G\), and \(k\). We obtain a triple \((A_i, \mathcal{P}_i, B_i)\) consisting of an arc set \(A_i \subseteq V(G) \times V(G)\), an \(s_i,t_i\)-maxflow \(P_i\) in \(G + A_i\), and a partition \(B_i\) of the deletable arcs on \(P_i\). For each \(P_i\) let \(k_i\) be its flow value and fix an arbitrary ordering \(P_{1i}, P_{2i}, \ldots, P_{ki}\) of the paths in \(P_i\). Note that not necessarily \(k_i \leq k\), but this is the case if \(A_i\) is in a sense compatible with a solution, see the notion of safe augmentation below. Hence, at this point if for some \(i \in [3]\) we have \(k_i > k\) we stop and return a failure symbol. Otherwise we continue and denote \(G_i := G + A_i\).

In order to define a Permutation CSP instance, for each \(i \in [3]\) and \(j \in [k_i]\) we introduce a variable \(x^i_j\) with domain \(D^i_j := V(P^i_j) \setminus \{(s_i,t_i) \cup V^\infty\}\) and a variable \(x'^i_j\) with domain \(D'^i_j := V(P^i_j) \setminus \{(s_i,t_i) \cup V^\infty\}\). The ordering \(\leq^i_j\) of \(D^i_j\) corresponds to a traversal of the path \(P^i_j\) from \(s_i\) to \(t_i\) and the ordering \(\leq'^i_j\) of \(D'^i_j\) corresponds to a traversal of the path \(P^i_j\) from \(t_i\) to \(s_i\). (Intuitively, \(x^i_j\) and \(x'^i_j\) always choose the same vertex, but we need both orderings of their
domain in order to define downwards-closed constraints below.) Below we let \( X \) denote the set of variables \( x_j^i \) and \( X' \) the set of variables \( x_{j'}^i \).

As to the constraints, first, for each \( i \in [3] \) and \( j \in [k] \) we introduce a permutation constraint \( \rho_j^i : D_j^i \to D_j^h \) that ensures that \( x_j^i \) and \( x_{j'}^i \) are the same, that is, for each \( u \in D_j^i \) let \( \rho_j^i(u) = u \).

Next, we define the constraints that ensure that for each \( i \in [3] \) the vertices selected by the variables form an \( s_t_t_i \)-separator. For each \( i \in [3] \) and each pair of variables \( x_j^i \in X, x_{j'}^i \in X' \) introduce a constraint \( R_{j,j'}^i \subseteq D_j^i \times D_j^h \) as follows; refer to Figure 2 for an illustration. At first, put \( R_{j,j'}^i = D_j^i \times D_j^h \). Then, for each \( u \in V(P_j^i) \) and \( v \in V(P_{j'}^i) \) such that there is a path \( Q \) in \( G_i \) from \( u \) to \( v \) such that \( Q \) is internally vertex-disjoint from each path \( P_j^j \in P_i \), remove from \( R_{j,j'}^i \) all pairs \((x, y)\) such that \( u \leq_j x \) and \( v \leq_{j'} y \). (This means that \( u \) occurs before \( x \) on path \( P_j^i \) and \( v \) occurs after \( y \) on path \( P_{j'}^i \). Intuitively, no solution may choose \( x \) and \( y \) because \( Q \) bypasses the corresponding vertex set, showing it is not a separator. Note that the path \( Q \) may consist of a single edge and that one of its endpoints may be \( s \) or \( t \).) This finishes the description of the constraint \( R_{j,j'}^i \). (Note that not necessarily \( R_{j,j'}^i = R_{j,j'}^j \).) Note that if \((x, y) \in R_{j,j'}^i \) and \( x' \leq_j x \) and \( y' \leq_{j'} y \) then \((x', y') \in R_{j,j'}^i \): Otherwise, at the point where we have removed \((x', y')\) from \( R_{j,j'}^i \) in the above construction, say due to some pair \((u, v)\), we have \( u \leq_j x' \leq_j x \) and \( v \leq_{j'} y' \leq_{j'} y \) and thus we would have removed \((x, y)\) as well, a contradiction. Thus, \( R_{j,j'}^i \) is downwards-closed.

We add further constraints to the PERMUTATION CSP instance below. But first we observe that already at this point, every solution to the 3-DMC instance \( \langle G, k, (s, t)_{i \in [3]}, V^\infty \rangle \) induces a solution for the PERMUTATION CSP instance and every solution to the PERMUTATION CSP instance induces three separators between the three terminal pairs. Let \( C_1 \) denote the PERMUTATION CSP instance constructed so far, that is,

\[
C_1 = (X \cup X', (D_j^i \leq_j D_j^h, \leq_{j'}^{h_i})_{i \in [3], j \in [k]}, (\rho_j^i)_{i \in [3], j \in [k]}, (R_{j,j'}^i)_{i \in [3], j, j' \in [k]}).
\]

Observe that we can construct \( C_1 \) in polynomial time because the algorithm of Theorem 2.7
runs in polynomial time and the paths underlying the construction of \( R^i_{j,j'} \) can be computed in polynomial time.

For proving the soundness of the algorithm, we need the following definition. Let \( S \) be a solution to \((G,k,(s_i,t_i))_{i \in [3]}, V^\infty\). Define the event safely augmented (wrt. \( S \)) as the intersection of the three events \( E_i \), \( i \in [3] \), that state that there exists a minimal \( s_it_i \)-separator \( S_i \) that is contained in \( S \) such that \((A_i, P_i)\) is compatible with \( S_i \). We also say that the \( S_i \) are the witnesses to having safely augmented. Note that, if we have safely augmented, then the flow value \( k_i \) of each \( P_i \) is at most \( k \). In order to prove soundness, we make use of the following two statements.

**Lemma 4.1.** Let \( S \) be an arbitrary fixed solution to \((G,k,(s_i,t_i))_{i \in [3]}, V^\infty\). Then with probability at least \( 2^{-O(k^4 \log k)} \) we have safely augmented wrt. \( S \).

**Proof.** Let \( S_i \), \( i \in [3] \), be a minimal \( s_it_i \)-separator (in \( G \)) contained in \( S \). The three events \( E_i \) are independent from each other. They each have probability lower bounded by the probability of the event that \((A_i, P_i)\) is compatible with \( S_i \). By [Theorem 2.7](#) they thus each have probability at least \( 2^{-O(k^4 \log k)} \), which implies the desired bound. \( \square \)

**Lemma 4.2.** Assume we have safely augmented and let \( S_i \), \( i \in [3] \), be witnesses to that fact. Then, for each \( i \in [3] \) and for each \( j \in [k_i] \) we have \(|V(P^i_j) \cap S_i| = 1 \). Define the mapping \( \phi \) by defining \( \phi(x^i_j) \) and \( \phi(x'^i_j) \) both as the single vertex in \( V(P^i_j) \cap S_i \) for each \( i \in [3] \) and \( j \in [k_i] \). Then \( \phi \) is a solution to the Permutation CSP instance \( \hat{C}_1 \).

**Proof.** Fix \( i \in [3] \). Since we have safely augmented, \((A_i, P_i)\) is compatible with \( S_i \). Thus \( P_i \) is an \( s_it_i \)-maxflow of value \(|S_i|\), showing that for each \( j \in [k_i] \) we have \(|V(P^i_j) \cap S_i| = 1 \). We define \( \phi(x^i_j), \phi(x'^i_j) \) as specified in the statement. Clearly the permutation constraints \( p^i_j \) are satisfied. Assume towards a contradiction that some downwards-closed constraint \( R^i_{j,j'} \) is not satisfied. Thus, at some point in the construction of \( R^i_{j,j'} \) we have removed \((\phi(x^i_j), \phi(x'^i_j))\) from \( R^i_{j,j'} \), say due to some path \( Q \) from the vertex \( u \in V(P^i_j) \) to the vertex \( v \in V(P^i_{j'}) \). By the construction of \( R^i_{j,j'} \), we have \( u \preceq^i_j \phi(x^i_j) \) and \( v \preceq^i_{j'} \phi(x'^i_j) \). We construct an \( s_it_i \)-path \( Q' \) in \( G_i \) by following \( P^i_j \) from \( s_i \) to \( u \), then adding \( Q \), and finally following \( P^i_{j'} \) from \( v \) to \( t_i \). Observe that \( V(Q') \cap S_i = \emptyset \). Since \( A_i \) is compatible with \( S_i \), this implies that there is an \( s_it_i \)-path in \( G \), a contradiction to the fact that \( S_i \) is an \( s_it_i \)-separator. \( \square \)

In order to prove completeness, we need the following.

**Lemma 4.3.** If \( \phi \) is a solution to the Permutation CSP instance \( C_1 \), then for each \( i \in [3] \) the set \( \{ \phi(x^i_j) \mid j \in [k_i] \} \) is an \( s_it_i \)-separator in \( G \).

**Proof.** Fix \( i \in [3] \) and let \( S_i = \{ \phi(x^i_j) \mid j \in [k_i] \} \). We show that \( S_i \) is an \( s_it_i \)-separator in \( G_i \). Since \( G_i \) is a supergraph of \( G \) the statement then follows. For a contradiction, assume that there is an \( s_it_i \)-path \( Q \) in \( G_i - S_i \). Observe that \( Q \) has at least one internal vertex that is contained in a path \( P^i_j \); otherwise, \( P^i_j \) would not be an \( s_it_i \)-maxflow in \( G_i \). Divide \( Q \) into segments \( Q_1, Q_2, \ldots \) such that each segment is of maximal length with respect to not having an internal vertex of a path \( P^i_j \), \( j \in [k_i] \). That is, the endpoints of the segments are either \( s_i, t_i \), or an internal vertex of a path \( P^i_j \). Label each segment \( Q_p \) by a label in \( \{-,+\} \times \{-,+\} \) as follows. Let \( u, v \) be the first and the last vertex of \( Q_p \), respectively, and observe that one of them occurs as internal vertex on a path in \( P_i \). The first part of the label of \( Q_p \) is \( - \) if \( u = s_i \) or if \( u \) occurs on a path \( P^i_j \) before \( \phi(x^i_j) \); otherwise the first part of the label is \( + \). The second part of the label is \( + \) if \( v = t_i \) or if \( v \) occurs on a path \( P^i_j \) after \( \phi(x'^i_j) \); otherwise the second part of the label is \( - \). Note that, since \( Q \) is an \( s_it_i \)-path, there is at least one segment with label \( (-,+) \), say segment \( Q_{p} \). Let \( u, v \) be the first and the last vertex of \( Q_{p} \), respectively. Without loss of generality, by symmetry, assume that \( u \neq s_i \), that is, \( u \) appears as internal vertex on a path in \( P_i \), say \( P^i_j \). If \( v \) appears as
internal vertex on a path in $\mathcal{P}$, then let $P'_j$, be that path; otherwise take $P'_j$ to be an arbitrary path in $\mathcal{P}$ different from $P'_j$. By construction of $Q_p$, we have $u \leq j \phi(x'_j)$ and $v \leq j \phi(x'_i)$. But then, by construction of $R'_{j,j'}$, we would have removed $(\phi(x'_j), \phi(x'_i))$ from $R'_{j,j'}$ due to the path $Q_p$, a contradiction. Thus, indeed $S_i$ is an $s,t,$-separator in $G_i$. ☐

Next, we aim to add the consistency constraints between variables mentioned in the outline above. To this end, we iterate over all possibilities of variables being assigned to the same vertex. More precisely, we iterate over all of the $2^{O(k \log k)}$ partitions $\mathcal{X}$ of the variable set $X$ into at most $k$ parts of size at most three such that no part contains two variables $x'_i$, $x'_j$ with $i \neq j$. (Intuitively, these capture all possibilities because no two paths in $\mathcal{P}$ share a vertex and thus no cluster of pairwise equal variables exceeds size three.) We call this the \textit{consistency iteration}.

Next, for each pair of variables $x'_i, x'_j$ in the same part in $\mathcal{X}$, restrict their domains to the set of shared vertices of $P'_i$ and $P'_j$, that is, replace both $D'_i$ and $D'_j$ by their intersection $D'_i \cap D'_j$. Perform analogous restrictions to the domains of the variables $x''_i, x''_j$. Omit the thereby invalidated bindings from the constraints $\rho'_i$ and $R'_{j,j'}$.

Finally, we introduce the permutation constraints enforcing the guessed consistency relation represented by $\mathcal{X}$. For every pair of variables $x'_i, x'_j$, contained in same part in $\mathcal{X}$, we introduce the constraint $\pi^{i,j}: D'_i \rightarrow D'_j'$ mapping each $u \in D'_i$ as $\pi^{i,j}(u) = x'_i$. This concludes the description of the reduction to Permutation CSP. Let $\mathcal{C}_2$ denote the resulting Permutation CSP instance

$$(X \cup X', (D'_i \leq j, D'_j \leq j), \rho'_i \in [3], j \in [k], (\pi^{i,j}) \in [3], j \in [k], (\pi^{i,j} \in [3], j \in [k], (\pi^{i,j} \in [3], j \in [k])).$$

Note that iterating over all possibilities for $\mathcal{X}$ can be done in $2^{O(k \log k)}$ time and thus constructing all the instances $\mathcal{C}_2$ takes $2^{O(k \log k)} \cdot n^{O(1)}$ time.

We can now extend Lemmas 4.2 and 4.3 to $\mathcal{C}_2$. To this end, let $S$ be a solution to $(G, k, (s_i, t_i)_{i \in [3]}, V^\infty)$ and assume we have safely augmented with witnesses $S_i$. We say that $\mathcal{X}$ complies (with $S$ and the witnesses $S_i$) if for each pair $i, i' \in [3], j \in [k_i], i' \in [k_{i'}]$ we have that $V(P'_i) \cap S_i = V(P'_i) \cap S'_i$ if $x'_i$ and $x'_j$ are both contained in the same part of $\mathcal{X}$.

\textbf{Lemma 4.4.} Let $S$ be a solution to $(G, k, (s_i, t_i)_{i \in [3]}, V^\infty)$ and assume we have safely augmented with witnesses $S_i$. Then, one of the partitions considered in the consistency iteration complies.

\textbf{Proof.} Since we have safely augmented and by Lemma 4.2, we have $|V(P'_i) \cap S_i| = 1$ for each $i \in [3]$ and for each $j \in [k_i]$. For every vertex $v \in S$ define a set of variables $X_v := \{x'_i | V(P'_i) \cap S_i = \{v\}\}$, which yields a complying partition $\mathcal{X} := \{X_v | v \in S\}$ of $X$. To see that $\mathcal{X}$ is considered in the consistency iteration observe that $|\mathcal{X}| \leq k$. Furthermore, for each $i \in [3]$ there are no two variables $x'_i, x'_j$ in the same part in $\mathcal{X}$, which proves the claim. ☐

\textbf{Lemma 4.5.} Let $S$ be a solution to $(G, k, (s_i, t_i)_{i \in [3]}, V^\infty)$, assume we have safely augmented with witnesses $S_i$ and assume that $\mathcal{X}$ complies. Then, we have $|V(P'_i) \cap S_i| = 1$ for each $i \in [3]$ and for each $j \in [k_i]$. Define the mapping $\phi$ by defining $\phi(x'_j)$ and $\phi(x''_j)$ both as the single vertex in $V(P'_i) \cap S_i$ for every $i \in [3]$ and $j \in [k_i]$. Then, $\phi$ is a solution to the Permutation CSP instance $\mathcal{C}_2$.

\textbf{Proof.} By Lemma 4.2, $|V(P'_i) \cap S_i| = 1$ and the mapping $\phi$ satisfies all constraints $\rho'_i$ and $R'_{j,j'}$. It remains to show that the values of $\phi$ have not been removed from the domains and that the permutation constraints $\pi^{i,j}$ are satisfied.

For the claim about the domains, fix some variable $x'_j$. Towards a contradiction, assume that $\phi(x'_j)$ was removed in the domain-restriction step. Then, there is some other variable $x''_j$, that is
in the same part of \( \mathcal{X} \) such that \( \phi(x^i_j) \notin D^i_j \). In other words, \( V(P^i_j) \cap S_i \neq V(P^i_j') \cap S'_j \). Since \( \mathcal{X} \) complies, \( x^i_j \) and \( x'^i_j \) are in different parts in \( \mathcal{X} \), a contradiction.

In order to show that the permutation constraints are satisfied, fix some constraint \( \pi^{i,i'}_{j,j'} \). By construction of \( \pi^{i,i'}_{j,j'} \), variables \( x^i_j \) and \( x'^i_j \) are in the same part of \( \mathcal{X} \). Since \( \mathcal{X} \) complies, \( V(P^i_j) \cap S_i = V(P^i_j') \cap S'_j \). Thus, by definition of \( \phi \) we have \( \phi(x^i_j) = \phi(x'^i_j) \), that is, \( \pi^{i,i'}_{j,j'} \) is satisfied.

Finally, we prove that solutions to \( \mathcal{C}_2 \) yield solutions to our 3-DMC instance.

**Lemma 4.6.** If \( \phi \) is a solution to the Permutation CSP instance \( \mathcal{C}_2 \), then the set \( \{ \phi(x^i_j) \mid i \in [3], j \in [k_i] \} \) is a solution to \( (G, k, (s_i, t_i)_{i \in [3]}, V^\infty) \).

**Proof.** Let \( S = \{ \phi(x^i_j) \mid i \in [3], j \in [k_i] \} \) and for each \( i \in [3] \) let \( S_i = \{ \phi(x^i_j) \mid j \in [k_i] \} \). Note that \( S = S_1 \cup S_2 \cup S_3 \). In comparison to \( \mathcal{C}_1 \), instance \( \mathcal{C}_2 \) contains only smaller domains and more constraints. Thus, the conclusion of Lemma 4.3 still holds for \( \mathcal{C}_2 \). Hence, each \( S_i \) is an \( s_i t_i \)-separator in \( G \). Thus it remains to show that \( |S| \leq k \). For this, observe that, due to the constraints \( \pi^{i,i'}_{j,j'} \), we have that for each part \( Y \in \mathcal{X} \) all the variables in \( Y \) have the same value. Since \( |\mathcal{X}| \leq k \), by construction of \( \mathcal{X} \) it follows that \( |S| \leq k \), as required.

We now continue working with \( \mathcal{C}_2 \); first reducing the twin-width of its constraints and then applying the algorithm from Section 3.

**Twin-width reduction by irrelevant vertices.** We next show how to bound the twin-width of the constraints in \( \mathcal{C}_2 \). First, observe that the twin-width of the constraints \( \rho^i_j \) is already bounded. Recall the definition of grid rank \( \text{gr} \) from Section 3.

**Lemma 4.7.** For each \( i \in [3] \) and \( j \in [k_i] \) we have \( \text{gr}(\text{Adj}(\rho^i_j)) \leq 1 \).

**Proof.** Let \( M = \text{Adj}(\rho^i_j) \). Observe that \( M \) is an anti-diagonal matrix. Each cell of \( M \) with at least two distinct rows or columns thus contains an entry of the anti-diagonal. Consider the upper-left cell \( C \) in a rank-\( k \) division of \( M \). For a contradiction, assume that \( k \geq 2 \). Then, \( C \) contains an entry of the anti-diagonal. Consider the cell \( C' \) that is south east of \( C \). Cell \( C' \) contains only zero entries, a contradiction.

The crucial constraints are the constraints \( \pi^{i,i'}_{j,j'} \), and they are indeed a priori not of bounded twin-width. However, using an irrelevant vertex argument, we can bound their twin-width.

**Lemma 4.8.** There exists a computable function \( h : \mathbb{N} \to \mathbb{N} \) and an algorithm that, given an instance \( (G, k, (s_i, t_i)_{i \in [3]}, V^\infty) \), the constraint \( \pi^{i,i'}_{j,j'} \) in \( \mathcal{C}_2 \) for \( i, i' \in [3] \) distinct, \( j \in [k_i] \), and \( j' \in [k_{i'}] \) and the augmented paths \( P^i_j \), \( P^{i'}_{j'} \) together with the partitions \( \mathcal{B}_i \) and \( \mathcal{B}_{i'} \) (from Theorem 2.7), certifies that \( \text{gr}(\text{Adj}(\pi^{i,i'}_{j,j'})) \leq h(k) \) or finds a vertex \( v \in D^i_j \cap D^{i'}_{j'} \) such that there is no shadowless solution \( S \) with \( v \in S \) and:

- all vertices before \( v \) on \( P^i_j \) do not reach \( t_i \) and all vertices after \( v \) on \( P^i_j \) are not reachable from \( s_i \) in \( G - S \);
- all vertices before \( v \) on \( P^{i'}_{j'} \) do not reach \( t_{i'} \) and all vertices after \( v \) on \( P^{i'}_{j'} \) are not reachable from \( s_{i'} \) in \( G - S \).

Moreover, the algorithm runs in fixed-parameter time with respect to \( k \).

The proof is given in Section 5.

Note that if the instance is safely augmented and \( \mathcal{X} \) complies with \( S \), then \( P^i_j \) contains a unique vertex of \( S_i \), \( P^{i'}_{j'} \) contains a unique vertex of \( S_{i'} \), and this is the same vertex. Lemma 4.8
returns $v$ that is guaranteed not to be the said vertex, so we can remove it from the domains of the variables corresponding to $P^t_j$ and $P^t_{j'}$. Formally, we use Lemma 4.8 as follows. We iterate over all consistency constraints. That is, for each $i,j,i',j' \in [3], j \in [k_i], and j' \in [k_{j'}]$, we consider the constraint $\pi_{i,j,i',j'}$ in $C_2$. We iteratively apply the algorithm of Lemma 4.8 to it. If it returns a vertex $v$, then we remove $v$ from both the domains $D^t_j$ and $D^t_{j'}$ of $x^j_i$ and $x^j_{i'}$, and repeat. If it returns that the grid rank is at most $h(k)$, we continue to the next constraint. Since we may drop at most $n$ vertices from a domain, this eventually leads to an empty domain for a variable, that is, a no-instance, or a situation in which for every constraint $\pi_{i,j,i',j'}$ in $C_2$ the matrix has grid rank bounded by $h(k)$. In the latter case, we apply Theorem 3.1 to solve $C_2$ in fixed-parameter time with respect to $k$.

We are now able to prove Theorem 1.1.

**Theorem 1.1.** **Directed Multicut** with three terminal pairs is fixed-parameter tractable when parameterized by the size of the cutset (with a randomized algorithm).

**Proof.** We claim that the algorithm described in this section solves $(G,k, (s_i, t_i)_{i \in [3]}, V^\infty)$ in fixed-parameter time with respect to $k$ with large-enough probability. By the arguments given throughout the section, the algorithm indeed runs in fixed-parameter time. If the algorithm returns a vertex set $S$, then $S$ is a solution to $(G,k, (s_i, t_i)_{i \in [3]}, V^\infty)$ by Lemma 4.6.

Assume now that there is a solution $S$ to $(G,k, (s_i, t_i)_{i \in [3]}, V^\infty)$. By Theorem 2.2, we may assume that $S$ is shadowless. By Lemma 4.1, with probability at least $2^{-O(k^4 \log k)}$ the algorithm safely augmented with respect to $S$. By Lemma 4.4, one of the considered $X$ complies with $S$. Thus, by Lemma 4.6, the Permutation CSP instance $C_2$ has a solution. By Lemma 4.8, $C_2$ maintains having a solution even after removing the vertices computed in Lemma 4.8 from the respective domains. Thus, the algorithm of Theorem 3.1 returns a solution, as required. □

5 Irrelevant vertex rule—Proof of Lemma 4.8

In this section, we give a tool that reduces the complexity of complicated permutation constraints in $C_2$. In fact, we argue that due to the shadow removal (as described in Section 6) and an improved version of the flow augmentation (Theorem 7.2), we obtain the following. If the permutation constraint has a high grid rank (gr), then we can explicitly find a vertex $v$ that cannot play the role of the solution vertex in the given permutation constraint.

First, we provide some very brief intuition. Shadow removal grants us some additional information about the reachability relation with respect to the terminals: For example, if some vertex cannot be reached from $s_1$ and $s_2$ after removal of a shadowless solution $S$, then it must be reachable from $s_3$. Symmetrically, if a vertex cannot reach $t_1$ and $t_2$, it has to reach $t_3$. Using the large grid rank, this additional reachability can be easily extended first along flow path $P^t_{j'}$ and then along flow path $P^t_j$, which gives a forbidden path $s_3t_3$-path. However, this does not prove the existence of such an $s_3t_3$-path in the original graph, as the flow paths might contain augmented edges. Therefore, we make use of the improved flow augmentation (Theorem 7.2) in order to still reach a contradiction. In particular, if $S$ satisfies the additional assumption stated in Lemma 4.8 (which still follows from the flow augmentation), the soybeans (introduced in Section 2.3) provide the needed connectivity without using the augmented edges. Unfortunately, their structure is (and has to be; see Section 7) a bit more complicated than just a simple path, but using additional shadow removal arguments, we are able to derive the existence of an $s_3t_3$-path and therefore, the desired contradiction.

Note one more complication: The soybeans may use vertices in other flow paths, but as they are vertex disjoint, at most $k$ of them can intersect $S$, so if we set the threshold high enough, we still reach the contradiction.
Lemma 4.8. There exists a computable function \( h : \mathbb{N} \to \mathbb{N} \) and an algorithm that, given an instance \((G,k, (s_i, t_i) \in [3]^t, V_\infty)\), the constraint \( \pi_{i,j}^{x,y} \) in \( C_2 \) for \( i,i' \in [3] \) distinct, \( j \in [k] \), and \( j' \in [k_i] \) and the augmented paths \( P_{i,j}^i \), \( P_{j,j'}^i \) together with the partitions \( B_i \) and \( B_{i'} \) (from Theorem 2.7), certifies that \( gr(\text{Adj}(\pi_{i,j}^{x,y})) \leq h(k) \) or finds a vertex \( v \in D_i \cap D_{i'} \) such that there is no shadowless solution \( S \) with \( v \in S \) and:

- all vertices before \( v \) on \( P_{i,j}^i \) do not reach \( t_i \) and all vertices after \( v \) on \( P_{i,j}^i \) are not reachable from \( s_i \) in \( G - S \);
- all vertices before \( v \) on \( P_{j,j'}^i \) do not reach \( t_{i'} \) and all vertices after \( v \) on \( P_{j,j'}^i \) are not reachable from \( s_{i'} \) in \( G - S \).

Moreover, the algorithm runs in fixed-parameter time with respect to \( k \).

Proof. For simplicity, denote \( P_j = P_{i,j}^i \) and \( P_{j'} = P_{j,j'}^i \). Without loss of generality suppose that \( i = 1 \) and \( i' = 2 \).

For a (sub)matrix \( A \) of \( \text{Adj}(\pi_{i,j}^{x,y}) \), we refer to the vertices that correspond to domains of \( A \) as the vertices corresponding to \( A \). Note that whenever there is a 1 in the (sub)matrix \( A \), it means that the corresponding two domain values are, in fact, the same vertex in \( G \).

Recall functions \( c \) and \( q \) of Theorem 2.7. We set \( \zeta := 2q(k,q(k,k+1)+1+k)+1 \). Let \( \rho \) be the bipartite Ramsey number for \((c(k))^2 \) colors and a monochromatic biclique of size \( 2\zeta \); that is, every edge coloring of \( K_{c(k)} \) with \((c(k))^2 \) colors contains a monochromatic copy of \( K_{2\zeta,2\zeta} \). We apply Theorem 2.5 to \( \text{Adj}(\pi_{i,j}^{1,2}) \) and \( \rho \) to either obtain that \( gr(\text{Adj}(\pi_{i,j}^{1,2})) \) is bounded by \( 2^O(\rho \log \rho) \) (and a computable function of \( k \)), or obtain a \( \rho \)-division of \( \text{Adj}(\pi_{i,j}^{1,2}) \) that has at least one 1 in every cell. In the first case, we return that \( gr(\text{Adj}(\pi_{i,j}^{1,2})) \) is bounded by a computable function of \( k \), in the second case we proceed further; we leverage the obtained \( \rho \)-division to obtain the desired irrelevant vertex \( v \).

We fix one such entry with value 1 for every \((p,q)\)-cell of the said division, which corresponds to a vertex, and call it the \((p,q)\)-representative. For every cell \((p,q)\), we color it with a pair \((B_1,B_2)\), where for \( i = 1,2 \) the set \( B_i \in B_i \) is the set containing the \((p,q)\)-representative in the application of Theorem 2.7 for the pair \( s_i, t_i \). Clearly, there are at most \((c(k))^2 \) colors. By the choice of \( \rho \), there is a coarser \( 2\zeta \)-division of \( \text{Adj}(\pi_{i,j}^{1,2}) \) such that we can choose a representative in every cell of this division such that all representatives are of the same color \((B_1,B_2)\).

In what follows, we only work with the latter division and representatives and hence use the name \((p,q)\)-representative for them. For a (sub)matrix \( A \) of \( \text{Adj}(\pi_{i,j}^{1,2}) \), we say representatives for the set of all \((p,q)\)-representatives within \( A \).

We choose \( v \) as the \((\zeta,\zeta)\)-representative. Now, towards a contradiction, suppose that \( v \in S \) and \( v \) splits \( P_j \) and \( P_{j'} \) as in the lemma statement.

This split in particular applies to the representatives. The first part of \( P_j \) (resp. \( P_{j'} \)) contains the vertices not reaching \( t_1 \) (resp. \( t_2 \)) up to the cell index \( i = \zeta - 1 \) (resp. \( j = \zeta - 1 \)) and the second part contains vertices not reachable from \( s_1 \) (resp. \( s_2 \)) in \( G - S \). Based on the above, we split the cells of the matrix \( \text{Adj}(\pi_{i,j}^{1,2}) \) into four quadrants and take submatrices consisting only of their representatives: \( A_{112}, A_{112}, A_{121}, \) and \( A_{12} \), where the subscript indicates the non-reachability of the quadrant. More formally, \( A_{112} \) consists of the \((p,q)\)-representatives for \( 1 \leq p, q \leq \zeta - 1 \) and they do not reach \( t_1 \) nor \( t_2 \) in \( G - S \), \( A_{112} \) consists of the \((p,q)\)-representatives for \( 1 \leq p \leq \zeta - 1 \) and \( \zeta + 1 \leq q \leq 2\zeta \) (and they do not reach \( t_1 \) and are not reachable from \( s_2 \) in \( G - S \)), \( A_{121} \) consists of the \((p,q)\)-representatives for \( 1 \leq q \leq \zeta - 1 \) and \( \zeta + 1 \leq p \leq 2\zeta \) (and they do not reach \( t_2 \) and are not reachable from \( s_1 \) in \( G - S \)), and finally \( A_{12} \) consists of the \((p,q)\)-representatives for \( \zeta + 1 \leq p, q \leq 2\zeta \) (and they are not reachable from \( s_1 \) nor \( s_2 \) in \( G - S \)).

As \( S \) is shadowless thus all vertices in \( A_{112} \) have to reach \( t_3 \) and all vertices in \( A_{12} \) have to be reachable from \( s_3 \) in \( G - S \). In order to make use of the soybeans given by the flow-augmentation (Theorem 2.7) we need to define suitable interlaced sets. Let \( \xi \) be the vertices of \( P_{j'} \) corresponding to \((p,q)\)-representatives of \( A_{12} \) for every odd \( p \geq \zeta + 1 \) and any one fixed
\[ q \geq \zeta + 1, \text{ and let } \xi_v \text{ be the set of vertices of } P_j \text{ corresponding to } (p, q)\text{-representatives of } A_{11}\v for every even } p \geq \zeta + 1 \text{ and } q := p - \zeta. \text{ It follows that } \xi_v \text{ and } \xi_e \text{ form interlaced sets on path } P_j \text{ of size}

\[
\left\lfloor \frac{\zeta}{2} \right\rfloor = \lfloor (2q(k, q(k, k + 1) + 1 + k) + 1)/2 \rfloor \geq q(k, q(k, k + 1) + 1 + k).
\]

Hence we obtain \( q(k, k + 1) + 1 + k \) \( \xi_v, \xi_e \)-soybeans by Theorem 2.7 (note that the interlaced sets are within one set \( B_2 \in B_2 \)). Let \( Q_1Q_2 \) be one of these \( \xi_v, \xi_e \)-soybean that does not intersect \( S \). Pick two vertices \( v_1 \in \xi_v \) and \( v_2 \in \xi_e \). As \( v_1 \) corresponds to a vertex of \( A_{12} \), vertex \( v_1 \) is not reachable from \( s_1 \) and \( s_2 \) in \( G \). As \( Q_1Q_2 \) is disjoint with the solution \( S \), \( s(Q_1Q_2) \) is not reachable from \( s_1 \) nor \( s_2 \) in \( G - S \). Since \( S \) is shadowless, \( s(Q_1Q_2) \) is reachable from \( s_3 \) in \( G - S \). Again as \( Q_1Q_2 \) is disjoint with the solution \( S \), \( v_2 \) is reachable from \( s_3 \) in \( G - S \).

For an illustration, consider the left part of Figure 3. Some of the \( \xi_v, \xi_e \)-soybeans may intersect \( S \), but as they are pairwise vertex disjoint, at most \( k \) of them can. Hence, the above properties hold for \( q(k, k + 1) + 1 \) of them. We restrict the submatrix \( A_{11}\v \) only to entries in \( \xi_v \) that correspond to \( \xi_e, \xi_e \)-soybeans that are disjoint from \( S \). We call the resulting submatrix \( A_{11,2} \). Observe that \( A_{11,2} \) has at least \( q(k, k + 1) + 1 \) columns with at least one 1.

Now, we construct another pair of interlaced sets. Let \( \chi_1 \) be the set of vertices that are contained in \( \xi_e \) and that correspond to the soybeans selected in the previous step (i.e., to the representatives in \( A_{11,2} \), except for the last one on the path \( P_j \)). Note that by our previous construction, for \( (p, q)\text{-representatives in } \chi_1 \), all values of \( q \) are pairwise different and of the same parity. For every \( (p, q)\text{-representative in } \chi_1 \), we insert into \( \chi_2 \) a \((p', q')\text{-representative of } A_{12} \) for \( q' = q + 1 \) and any fixed \( p \leq \zeta - 1 \). It follows that \( \chi_1, \chi_2 \) are interlaced sets on path \( P_j \) of size \( q(k, k + 1) \), and they are within the same set \( B_k \in B_1 \). (We removed the last representative in \( \chi_1 \) as it could be the in the \((p, \zeta - 2)\text{-cell of } \text{Adj}(n_{j,j'}^{12}) \text{ for some } p \geq \zeta + 1 \).) Hence we obtain \( k + 1 \) vertex-disjoint \( \chi_1, \chi_2 \)-soybeans by Theorem 2.7. At most \( k \) of them can intersect \( S \). Therefore let \( Q_3Q_4 \) be one \( \chi_1, \chi_2 \)-soybean that does not. We know that \( v_3 \in Q_3 \cap \chi_1 \) is reachable by \( s_3 \) in \( G - S \) by the arguments in the previous paragraph.

We know that \( v_4 \in Q_4 \cap A_{11,2} \) does not reach \( t_1 \) nor \( t_2 \) in \( G - S \). Hence, \( t(Q_3Q_4) \) does not reach \( t_1 \) nor \( t_2 \). Since \( S \) is shadowless, \( t(Q_3Q_4) \) reaches \( t_3 \) in \( G - S \). Since \( Q_3Q_4 \) is disjoint with the solution, \( v_3 \) reaches \( t_3 \) in \( G - S \). This is the desired contradiction, see the right part of Figure 3 for this case.

\[ \square \]

6 Shadow removal

In this section, we prove Theorem 2.2. Recall that \((G, k, (s_i, t_i)_{i \in \llbracket 3 \rrbracket}, V^\infty)\) is an instance of 3-DMC. A set \( W \subseteq V(G) \) is called thin if for every \( v \in W \), \( v \notin \chi_{\text{adj}}(W \setminus v) \). For two sets \( A, B \subseteq V(G) \) such that \( A \cap B = \emptyset \), an \((A, B)\text{-separator} \) is a set \( S \subseteq V(G) \setminus (A \cup B) \) such that \( G - S \) has no path from any vertex of \( A \) to any vertex of \( B \). Let \( R^+_{G}(A) \) be the set of vertices that are reachable from some vertex of \( A \) in \( G \). An \((A, B)\text{-important separator} \) is an inclusionwise minimal \( S \subseteq V(G) \setminus (A \cup B) \) such that \( G - S \) has no path from \( A \) to \( B \) and there is no \((A, B)\text{-separator} S' \) such that \( |S'| \leq |S| \) and \( R^+_{G-S}(A) \subset R^+_{G-S}(A) \). To prove Theorem 2.2, we use the tool of random sampling of important separators from [24, 9, 8], presented in its derandomized form and in the form that is convenient for us to use, as Proposition 6.1.

**Proposition 6.1** (Theorem 3.18 by Chitnis et al. [8]). Given an instance \((G, k, (s_i, t_i)_{i \in \llbracket 3 \rrbracket}, V^\infty)\), there is a deterministic algorithm that runs in time \( 2^{O(k^2)} \cdot n^{O(1)} \) and outputs a family \( Z \subseteq 2^{|V(G)|} \) \( |V^\infty| \) of size \( 2^{O(k^2)} \log n \) such that the following holds. Let \( W \subseteq V(G) \) be a thin set of size at most \( k \) and let \( Y \subseteq V(G) \) such that for each \( v \in Y \), there is an important \((v, T)\text{-separator} \) \( W' \subseteq W \). For every such pair \((W, Y)\), there exists \( Z \in Z \) such that \( Z \cap W = \emptyset \) and \( Y \subseteq Z \).
Figure 3: The left part shows a basic layout of the matrix $\text{Adj}(\pi_{1}^{1,2})$. Each cell might contain multiple 1 entries, but we show only the representatives in the picture. The vertex $v$ is marked by a red circle. The submatrices $A_{t12}$, $A_{t1s2}$, $A_{t2s1}$, and $A_{s12}$ have a light-green background. The interlaced sets $\xi_o$ and $\xi_e$ on $P_{j'}$ are marked in blue and orange. One $\xi_o\xi_e$-soybean is depicted using dotted lines. If such a soybean does not intersect $S$, we derive the conclusions stated in red. The right part shows the interlaced sets $\chi_1$ (marked orange) and $\chi_2$ (marked purple) on $P_{j'}$. Note that some filtering steps have to happen in the proof, so only vertices for which we derived the conclusion (Reachable from $s_3$!) are part of $\chi_1$. One $\chi_1\chi_2$-soybean is depicted using dotted lines. If such a $\chi_1\chi_2$-soybean does not intersect $S$, we derive the conclusions stated in red and, therefore, a contradiction.
The idea is to use Proposition 6.1 to find a family of sets such that at least one of the sets in this collection contains the reverse and forward shadow of some solution, but does not contain the corresponding solution. We apply Proposition 6.1 in two steps: first to cover the reverse shadow of a solution and then to also cover its forward shadow. To apply Proposition 6.1 where the set $W$ corresponds to some solution and the set $Y$ corresponds to the reverse shadow of $W$, one needs to guarantee that there is a solution $W$ and its reverse shadow $Y$ that satisfies the properties of Proposition 6.1. In order to prove this, we define shadow-maximal solutions (as in [9, 8]).

**Definition 6.2** (Shadow-maximal solution). Let $I = (G, k, (s_i, t_i)_{i \in [\infty]}, V)$ be an instance of 3-DMC. An inclusion-wise minimal solution $S$ for $I$ is called a shadow-maximal solution if $r_G(S) \cup f_G(S) \cup S$ is inclusion-wise maximal among all minimal solutions $S$.

A shadow-maximal solution $S$ is called a best shadow-maximal solution, if it is shadow-maximal and amongst all shadow-maximal solutions $|r_G(S)|$ is maximum.

We now show in Lemmas 6.3 and 6.4 that if $W$ is a best shadow-maximal solution and $Y = r_G(W)$, then $W$ and $Y$ satisfy the properties of Proposition 6.1.

**Lemma 6.3.** Every minimal solution of the instance $(G, k, (s_i, t_i)_{i \in [\infty]}, V)$ of 3-DMC is thin.

*Proof.* Let $W$ be a minimal solution of the instance $(G, k, (s_i, t_i)_{i \in [\infty]}, V)$. We show that if $v \in W$ and $v \in r_G(W')$ for some $W' \subseteq W$, then $W' \setminus v$ is also a solution for $(G, k, (s_i, t_i)_{i \in [\infty]}, V)$, contradicting the minimality of $W$. Towards this, if $W' \setminus v$ is not a solution, then there exists an $s_i t_j$-path in $G - (W \setminus v)$ that contains $v$. This implies the existence of a $v t_j$-path, call it $P$, in $G - (W \setminus v)$. Since $v \in r_G(W')$, $P$ must contain a vertex of $W' \subseteq W \setminus v$, which yields a contradiction. \qed

**Lemma 6.4.** Let $(G, k, (s_i, t_i)_{i \in [\infty]}, V)$ be an instance of 3-DMC and let $S$ be a shadow-maximal solution. Then, either for every $v \in r_G(S)$, there exists $S_v \subseteq S$ such that $S_v$ is a $(v, \{t_1, t_2, t_3\})$-important separator, or there exists another shadow-maximal solution $S'$ such that $r_G(S) \subseteq r_G(S')$.

*Proof.* Fix $v \in r_G(S)$. Then $S$ is a $(v, \{t_1, t_2, t_3\})$-separator in $G$. Let $S_v \subseteq S$ be a minimal $(v, \{t_1, t_2, t_3\})$-separator. If $S_v$ is an important $(v, \{t_1, t_2, t_3\})$-separator then we are done. Otherwise there exists a minimal $(v, \{t_1, t_2, t_3\})$-separator $S'_v$ such that $|S'_v| \leq |S_v|$ and $r_{G-S_v}(v) \subseteq r_{G-S'_v}(v)$. Let $S' = (S \setminus S_v) \cup S'_v$. Clearly $|S'| \leq |S|$ since $|S'_v| \leq |S_v|$. Also, since $S'_v \neq S_v$, we have $S \setminus S' \neq \emptyset$. We now show that $S'$ is a solution such that $(r_G(S) \cup f_G(S) \cup S) \subseteq (r_G(S') \cup f_G(S') \cup S')$ and $r_G(S) \subseteq r_G(S')$, which proves the lemma.

**Claim 6.5.** $r_G(S) \cup (S \setminus S') \subseteq r_G(S')$.

*Proof.* We first show that if $x \in S \setminus S' (= S_v \setminus S'_v)$, then $x \in r_G(S')$. Suppose not (that is $x \notin r_G(S')$). Then there exists an $(x, \{t_1, t_2, t_3\})$-path in $G - S'$. Since $S_v$ is a minimal $(v, \{t_1, t_2, t_3\})$-separator and $x \in S_v$, there exists a $(v, \{t_1, t_2, t_3\})$-path that intersects $S_v$ exactly at $x$. Let $P$ denote the $v x$-subpath of this path. Then $V(P) \setminus x \subseteq R_{G-S_v}^+(v)$. Since $S'_v$ is a $(v, \{t_1, t_2, t_3\})$-important separator and $x \notin S'_v$, $V(P) \subseteq R_{G-S'_v}^+(v)$. Thus, there is a $v x$-path in $G - S'$. This, together with the $(x, \{t_1, t_2, t_3\})$-path in $G - S'$, implies a $(v, \{t_1, t_2, t_3\})$-path in $G - S'$. Since $S'_v \subseteq S'$ and $S'_v$ is a $(v, \{t_1, t_2, t_3\})$-separator, this is not possible.

Now suppose, for the sake of contradiction, that $x \in r_G(S)$ but $x \notin r_G(S')$. Consider an $(x, \{t_1, t_2, t_3\})$-path $P$ in $G - S'$. Then there exists $y \in V(P)$ such that $y \in S \setminus S'$. From the claim in the previous paragraph, $y \in r_G(S')$. Thus, there exists a vertex of $S'$ on the $(y, \{t_1, t_2, t_3\})$-subpath of $P$, which is a contradiction. \qed
From Claim 6.5 and since $|S \setminus S'| \geq 1$, $r_G(S) \subseteq r_G(S')$. We now show that $S'$ is a solution for the instance $(G, k, (s_i, t_i)_{i \in [3]}, V^\infty)$.

**Claim 6.6.** $S'$ is a solution for the instance $(G, k, (s_i, t_i)_{i \in [3]}, V^\infty)$ of 3-DMC.

**Proof.** For the sake of contradiction, say $S'$ is not a solution. Then there exists an $s_i t_j$-path in $G - S'$ that uses a vertex $x \in S_v \setminus S'_v$. In particular, there exists an $x t_j$-path in $G - S'$, that is $x \not\in r_G(S')$. Since $S_v \cap S'_v = S \setminus S'$, from Claim 6.5, $x \in r_G(S')$. This is a contradiction. \qed

**Claim 6.7.** $(r_G(S) \cup f_G(S) \cup S) \subseteq (r_G(S') \cup f_G(S') \cup S')$.

**Proof.** From Claim 6.5, $(r_G(S) \cup S) \subseteq (r_G(S') \cup S')$. We now show that for any $v \in f_G(S) \setminus r_G(S)$, $v \in f(S') \cup r_G(S')$. Suppose not. Then there exists an $s_i v$-path, say $P_1$, and a $v t_j$-path, say $P_2$, for some $i, j \in [3]$, in $G - S'$. Since $v \not\in f_G(S)$, the path $P_1$ contains a vertex $x \in S \setminus S'$. From Claim 6.5, $x \in r_G(S')$. But the $x v$-subpath of $P_1$, together with the $v t_j$-path $P_2$ gives an $x t_j$-path in $G - S'$, which is a contradiction because $x \in r_G(S')$. \qed

From Claims 6.5 and 6.6 and since $|S \setminus S'| \geq 1$, we have shown that $S'$ has the properties stated earlier, which contradict that $S$ is a best shadow-maximal solution. \qed

The following lemma, uses Lemmas 6.3 and 6.4 to show that appropriate applications of Proposition 6.1 result in a family of sets that cover the shadow of some solution.

**Lemma 6.8** (Covering the shadow). There is an algorithm that given an instance $I = (G, k, (s_i, t_i)_{i \in [3]}, V^\infty)$ of 3-DMC, in time $2^{O(k^2)} \cdot n^{O(1)}$, outputs a collection $Z \subseteq 2^{V(G)} \setminus V^\infty$ of size $2^{O(k^2)} \log^2 n$, such that if $I$ is a Yes-instance, then there exists a solution $S^*$ such that there exists $Z \in Z$ for which $Z \cap S^* = \emptyset$ and $r_G(S^*) \cup f_G(S^*) \subseteq Z$.

**Proof.** Let $X$ be the family returned by the algorithm of Proposition 6.1 on the instance $I$. Suppose $I$ is a Yes-instance. Fix a best shadow-maximal solution $S^*$ of $I$. From Lemmas 6.3 and 6.4 follows that the pair of sets $(W, Y) = (S^*, r_G(S^*))$ satisfies the properties of the sets in Proposition 6.1 on input $I$. Thus, there exists $Z' \subseteq X$ such that $Z' \cap S^* = \emptyset$ and $r_G(S^*) \subseteq Z'$.

Let $\bar{G}$ be the graph obtained from $G$ after reversing all the edges of $G$. For each $Z \in X$, create an instance $I_Z = (\bar{G}, k, (t_i, s_i)_{i \in [3]}, V^\infty)$, where $V^\infty = V^\infty \cup Z$. Let $Y_Z$ be the family returned by the algorithm of Proposition 6.1 on input $I_Z$. Then output the family $Z = \{Z_1 \cup Z_2 : Z_1 \in X, Z_2 \in Y_Z\}$. We now show that $Z$ is the desired family. The size bound and the running time follow from Proposition 6.1.

Observe that $S^*$ is also a solution for $I_Z$. Further observe that for any set $S \subseteq V(G)$, the forward shadow of $S$ in $G$, with respect to $\{s_1, s_2, s_3\}$ is equal to the reverse shadow of $S$ in $\bar{G}$ with respect to $\{t_1, t_2, t_3\}$, and the reverse shadow of $S$ in $G$, with respect to $\{t_1, t_2, t_3\}$ is equal to the forward shadow of $S$ in $\bar{G}$ with respect to $\{s_1, s_2, s_3\}$. That is, $r_G(S) = f_{\bar{G}}(S)$ and $f_G(S) = r_G(S)$. In particular, $S^*$ is a shadow-maximal solution of $I_Z$. We now want to show that the pair $(W, Y) = (S^*, r_G(S^*))$ satisfies the properties of Proposition 6.1 on input $I_Z$. Towards this we prove the following claim.

**Claim 6.9.** For each $v \in r_G(S^*)$, there exists an $S_v \subseteq S^*$ that is a $(v, \{s_1, s_2, s_3\})$-important separator in $\bar{G}$.

**Proof.** Suppose the claim does not hold. Then, by Lemma 6.4, there exists a shadow-maximal solution $S'' \subseteq I_Z$, such that $r_{\bar{G}}(S'') \subset r_{\bar{G}}(S')$. Since $S'' \subseteq I_Z$ is also a shadow-maximal solution of $I_Z$, we conclude that $(r_{\bar{G}}(S'') \cup f_{\bar{G}}(S'') \cup S'') = (r_{\bar{G}}(S') \cup f_{\bar{G}}(S') \cup S')$. In particular, $S'' \setminus S' \subseteq (r_{\bar{G}}(S'') \cup f_{\bar{G}}(S'') \cup S'')$. That is, for any $v \in S'' \setminus S'$, either $v \in r_{\bar{G}}(S'')$ or $v \in f_{\bar{G}}(S'')$.

Since $f_{\bar{G}}(S'') = r_{\bar{G}}(S'') \subseteq Z' \subseteq V^\infty$, and $S''$ is a solution of $I_Z$, $S'' \cap f_{\bar{G}}(S'') = \emptyset$. If $v \in r_{\bar{G}}(S'')$, then, since $r_{\bar{G}}(S'') \subset r_{\bar{G}}(S')$, $v \in r_{\bar{G}}(S')$. This is a contradiction as $v \in S'' \setminus S'$. \qed
Thus from Claim 6.9 and Lemma 6.3 follows that the pair \((W,Y) = (S^*, \lambda_G(S^*))\) satisfies the properties of Proposition 6.1 on input \(I_G\). Thus, there exists \(Z' \in \mathcal{Y}_G\) such that \(Z' \cap S^* = \emptyset\) and \(f_G(S^*) \subseteq Z'\). Let \(Z^* = Z' \cup Z'\). Then from the above arguments \(Z^* \cap S^* = \emptyset\) and \((r_G(S^*) \cup f_G(S^*)) \subseteq Z^*\). Also \(Z^* \in \mathcal{Z}\).

Finally we use Lemma 6.8 to prove Theorem 2.2.

**Theorem 2.2.** Given an instance \((G,k, (s_i, t_i)_{i \in [3]}, V^\infty)\) of 3-DMC, there is an algorithm that runs in time \(2^{O(k^2 \log k)} n^{O(1)}\), and outputs a family \(Z\) of subsets of \(V(G) \setminus V^\infty\) such that \(|Z| = 2^{O(k^2 \log k)}\) and, if the input instance is a \(\text{Yes}\)-instance, then there exists \(Z \in \mathcal{Z}\) such that \((G',k, (s_i, t_i)_{i \in [3]}, V^\infty \setminus Z)\) is a \(\text{Yes}\)-instance that admits a shadowless solution of cardinality at most \(k\), where \(G'\) is the result of bypassing \(Z\) in \(G\).

**Proof.** Given an instance \((G,k, (s_i, t_i)_{i \in [3]}, V^\infty)\), let \(Z\) be the collection returned by Lemma 6.8. From Lemma 6.8, if \(Z\) is a \(\text{Yes}\)-instance, then there exists a solution \(S\) and \(Z \in \mathcal{Z}\) such that \(Z \cap S = \emptyset\) and \((r_G(S) \cup f_G(S)) \subseteq Z\). Let \(G'\) be obtained from \(G\) by bypassing \(Z\). From Lemma 2.1, \(S\) is also a solution of \((G',k, (s_i, t_i)_{i \in [3]}, V^\infty \setminus Z)\). We now show that \(S\) is a shadowless solution of \((G',k, (s_i, t_i)_{i \in [3]}, V^\infty)\), that is \(r_G(S) = f_G(S) = \emptyset\). For the sake of contradiction, say \(r_G(S) \neq \emptyset\). Let \(x \in r_G(S)\). In particular, \(x \in V(G') = V(G) \setminus Z\). Since \(x \in r_G(S)\), \(G - S\) has no \((x, \{t_1, t_2, t_3\})\)-path. From Lemma 2.1, \(G - S\) also has no \((x, \{t_1, t_2, t_3\})\)-path, which implies that \(x \in r_G(S)\). This is a contradiction, since then \(x \in Z\) and hence \(x \notin V(G')\).

7 Harvesting soybeans

This section is devoted to the proof of Theorem 2.7. The proof revisits the whole proof of flow-augmentation of [15] (recalled below as Theorem 7.1) and extracts the additional information along the way. Furthermore, we need to slightly revise the behavior of the algorithm in the base case to ensure the desired properties. This section assumes that the reader is familiar with the proof of Theorem 7.1 from [15].

7.1 Back to the edge-deletion regime

Since flow-augmentation in [15] is defined on edge-cuts, but Theorem 2.7 is in the vertex-deletion regime, we first go back to the edge-deletion regime. We need a few more definitions that closely follow [15].

In the edge-deletion regime, the edges of \(G\) can be deletable or undeletable. An (edge-based) \(st\)-flow, in this context, is a collection of \(st\)-paths that do not share a deletable edge and an \(st\)-cut is a set of deletable edges that intersects all \(st\)-paths. It is always clear from the context whether we speak about the vertex- or the edge-deletion regime, and hence we reuse names like \(st\)-flow or \(st\)-mincut or the notation \(\lambda_G(s,t)\) for both regimes.

An \(st\)-cut is a \(st\)-cut if for every \((u, v) \in Z\) in the graph \(G - Z\) there is a path from \(s\) to \(u\) but there is no path from \(s\) to \(v\); observe that every minimal \(st\)-cut is a \(st\)-cut. For a \(st\)-cut \(Z\) in \(G\), by \(\text{core}_G(Z) \subseteq Z\) we denote the set of arcs \((u,v) \in Z\) such that there exists a path from \(v\) to \(t\) in \(G - Z\).

Being compatible is slightly more complicated for star cuts: a set of arcs \(A \subseteq V(G) \times V(G)\) is compatible with a star \(st\)-cut \(Z\) if the set of vertices reachable from \(s\) in \((G - Z)\) is the same. We also need the notion of a witnessing flow: for a star \(st\)-cut \(Z\) in \(G\), if \(\text{core}(Z)\) is an \(st\)-mincut, then an \(st\)-maxflow \(P\) is a witnessing flow if \(E(P) \cap Z = \text{core}(Z)\). The pair \((A, P)\) is compatible with \(Z\) if \(A\) is compatible with \(Z\), \(\text{core}_{G+A}(Z)\) is an \(st\)-mincut in \(G + A\), and \(P\) is a witnessing flow for \(Z\) in \(G + A\). Note that if \(Z\) is a minimal \(st\)-cut, and \(A\) is compatible with
Z, then any st-maxflow in \( G + A \) is a witnessing flow for Z, but if Z is a star st-cut, the notion of a witnessing flow is more intricate.

The flow-augmentation technique is represented by the following statement.

**Theorem 7.1** (Kim et al. [15]). There exists a polynomial-time algorithm that, given a directed graph \( G \), vertices \( s,t \in V(G) \), and an integer \( k \), returns a set \( A \subseteq V(G) \times V(G) \) and an st-maxflow \( \mathcal{P} \) in \( G + A \) such that for every star st-cut \( Z \) of size at most \( k \), with probability \( 2^{-O(k^4 \log k)} \) the pair \((A,\mathcal{P})\) is compatible with \( Z \).

Most of this section is devoted to the proof of the following edge-deletion variant of Theorem 2.7.

**Theorem 7.2.** There exist computable functions \( c : \mathbb{N} \to \mathbb{N} \) and \( q : \mathbb{N} \times \mathbb{N} \to \mathbb{N} \) such that the following holds.

There exists a polynomial-time randomized algorithm that, given a directed graph \( G \), vertices \( s,t \in V(G) \), and an integer \( k \), returns a set \( A \subseteq V(G) \times V(G) \) and an st-maxflow \( \mathcal{P} \) in \( G + A \) such that for every star st-cut \( Z \) of size at most \( k \), with probability \( 2^{-O(k^4 \log k)} \), the pair \((A,\mathcal{P})\) is compatible with \( Z \).

Additionally, the algorithm returns a partition \( B \) of the deletable edges of \( \bigcup_{P \in \mathcal{P}} E(P) \) into at most \( c(k) \) sets such that for every \( P \in \mathcal{P} \), every integer \( p \in \mathbb{N} \), every \( B \in B \) and every two disjoint sets \( C,D \) of size at least \( q(k,p) \), consisting of edges of \( B \cap E(P) \) that are interlaced on \( P \), the graph \( G \) contains a family of \( p \) pairwise vertex-disjoint CD-soybeans.

Finally, one can take \( c \) and \( q \) such that \( c(k) = 2^O(k^3 \log k) \) and \( q(k,p) = 2^O(k^3 \log(kp)) \).

Observe that if \( P \cap A = \emptyset \), that is, \( P \) does not contain any augmentation edge, then one can simply enumerate \( C = \{c_1, \ldots, c_q\} \) and \( D = \{d_1, \ldots, d_q\} \) for \( q = q(k,p) \) along the path \( P \), define \( P_i \) to be the subpath of \( P \) from \( c_i \) to \( d_i \), and use soybeans \( \{P_{2i}, P_{2i+1} \mid 1 \leq i \leq q/2\} \). (We skip every second such soybean in order to make them vertex-disjoint.) However, when \( P \) contains augmentation edges, the situation is more complex, as a soybean cannot use them.

We now formally show that Theorem 7.2 implies Theorem 2.7, restated below:

**Theorem 2.7.** There exist computable functions \( c : \mathbb{N} \to \mathbb{N} \) and \( q : \mathbb{N} \times \mathbb{N} \to \mathbb{N} \) such that the following holds.

There exists a polynomial-time randomized algorithm that, given a directed graph \( G \) (with possibly some vertices marked as undeletable), vertices \( s,t \in V(G) \), and an integer \( k \), returns an arc set \( A \subseteq V(G) \times V(G) \) and an st-maxflow \( \mathcal{P} \) in \( G + A \) such that for every minimal st-separator \( Z \) of size at most \( k \), with probability \( 2^{-O(k^4 \log k)} \), the tuple \((A,\mathcal{P})\) is compatible with \( Z \).

Additionally, the algorithm returns a partition \( B \) of the deletable vertices of \( \bigcup_{P \in \mathcal{P}} V(P) \) into at most \( c(k) \) sets such that for every \( P \in \mathcal{P} \), every integer \( p \in \mathbb{N} \), every \( B \in B \) and every two disjoint sets \( C,D \) of size at least \( q(k,p) \), consisting of vertices of \( B \cap V(P) \) that are interlaced on \( P \), the graph \( G \) contains a family of \( p \) pairwise vertex-disjoint CD-soybeans.

Finally, one can take \( c \) and \( q \) such that \( c(k) = 2^O(k^3 \log k) \) and \( q(k,p) = 2^O(k^3 \log(kp)) \).

**Proof.** We construct a new graph \( G' \) as follows. For each vertex \( v \in V(G) \), graph \( G' \) contains two vertices \( v_1 \) and \( v_2 \) and a deletable arc \((v_1,v_2)\); we call these arcs important. If \( v \) is undeletable or if \( v = s \) or \( v = t \), then the arc \((v_1,v_2)\) is undeletable instead. For each arc \((u,v) \in A(G) \), graph \( G' \) contains an undeletable arc \((u_2,v_1)\).

We apply Theorem 7.2 to \( G' \), \( s_2 \), \( t_1 \), and \( k \), obtaining an arc set \( A' \), an edge-based \( s_2t_1 \)-maxflow \( \mathcal{P}' \) in \( G' + A' \), and a partition \( B' \) of deletable arcs of \( \mathcal{P}' \).

We now perform the following cleanup step of \( A' \) and \( \mathcal{P}' \). First, for every \( x,y \in V(G') \) such there is a path from \( x \) to \( y \) in \( G' + A' \) that uses undeletable arcs only (recall that every arc of \( A' \) is considered undeletable), we add an undeletable arc \((x,y) \) to \( A' \) if it is not present already.
in \( G' + A' \). Second, for every \( P' \in \mathcal{P}' \) and every maximal subpath \( Q \) of \( P' \) that uses undeletable arcs only and is of length at least 2, we replace \( Q \) with only an arc \((s(Q), t(Q))\) (which is now in \( G' + A' \) due to the previous step). Finally, we restrict \( A' \) to contain only arcs that are present on some flow path in \( \mathcal{P}' \).

Observe that the first step does not change the space of \( s_2 t_1 \)-cuts in \( G' + A' \) (as any newly added arc \((x, y)\) cannot lead from the \( s \)-side to the \( t \)-side of any \( s_2 t_1 \)-cut in \( G' + A' \) due to the assumed path of undeletable edges from \( x \) to \( y \)) while \( \mathcal{P}' \) remains an \( s_2 t_1 \)-maxflow in \( G' + A' \) throughout the process (as we only reroute the paths through undeletable arcs) and every path \( P' \in \mathcal{P}' \) still visits the same deletable arcs, in the same order (so the properties of \( \mathcal{B}' \) are unharmed). The deletion of arcs in \( A' \) in the last step only extends the space of \( s_2 t_1 \)-cuts in \( G' + A' \), so if \((A', \mathcal{P}')\) was compatible with some \( s_2 t_1 \)-cut \( Z \) before the process, then it is also compatible after the process. Furthermore, as the only deletable arcs in \( G' + A' \) are the important arcs, in the end we have the following property: every \((x, y) \in A'\) is of the form \((u_2, v_1)\) for some \( u, v \in V(G) \).

The above discussion implies that we can obtain the desired result \((A, \mathcal{P}, \mathcal{B})\) in the natural manner: for every \((x, y) = (u_2, v_1) \in A'\), we add \((u, v)\) to \( A\), for every \( P' \in \mathcal{P}' \) we add to \( \mathcal{P} \) a path \( P \) being the path \( P' \) with all important edges on it contracted, and for every \( B' \in \mathcal{B}' \) we add to \( \mathcal{B} \) the set \( B = \{v \mid (v_1, v_2) \in B'\} \).

Thus, it remains to prove Theorem 7.2. This proof spans the rest of this section.

### 7.2 Initial setup

The algorithm of Theorem 7.1 first filters out some trivial cases in which \( A = \emptyset \) can be returned. In these cases one can take \( \mathcal{B} \) to be a singleton and construct the desired soybeans as in the comment after the statement of Theorem 7.2. After treating these trivial cases, the algorithm of Theorem 7.1 invokes a recursive subroutine.

The recursion has depth strictly less than \( h_{\max}(k) = \mathcal{O}(k^3) \). The input to the recursive call consists of a graph \( G \) with distinguished vertices \( s, t \in V(G) \), an integer \( k \), an \( st \)-flow \( \mathcal{P} \), and an integer \( \kappa \). The output is a set \( A \subseteq V(G) \times V(G) \) and an \( st \)-maxflow \( \mathcal{P} \) such that \( \lambda_G, A(s, t) \geq \kappa \) and for every star \( \text{st-cut} \) \( Z \) with \( |Z| \leq k \) and \( |\text{core}(Z)| \geq \kappa \), \((A, \mathcal{P})\) is compatible with \( Z \) with high probability.

An important insight about the structure of the recursion is that for every recursive call \( \rho \) on a graph \( G \) the following holds: For every subcall on a graph \( G', G' - \{s, t\} \) is a subgraph of \( G - \{s, t\} \). Furthermore, the graphs \( G' - \{s, t\} \) over all recursive subcalls are pairwise vertex-disjoint (as subgraphs of \( G \)).

**Constructing \( \mathcal{B} \).** Let \((A, \mathcal{P})\) be the output of the algorithm of Theorem 7.1 and consider \( P \in \mathcal{P} \). (We separately partition the deletable edges of \( E(P) \) into sets of \( \mathcal{B} \), so that every \( B \in \mathcal{B} \) is contained in \( E(P) \) for one \( P \in \mathcal{P} \).) If present, we put at most two deletable edges of \( E(P) \) that are incident with \( s \) or \( t \) into separate singleton sets of \( \mathcal{B} \) and do not worry about them further.

For every vertex \( v \in V(G) \), we can consider all recursive calls on graphs \( G' \) that contain \( v \). By the previous observation on the structure of the recursion, these calls form an upward path in the recursion tree. For every call \( \rho' \) in the recursion whose parent \( \rho \) corresponds to the “large \( \ell' \)” case, we assign to \( e \) a local signature as follows. The recursive call is applied to a graph \( G^\alpha \) for some \( \alpha \) such that \( G^\alpha - \{s, t\} \) is a subgraph of \( G \) between the \( st \)-mincuts \( C_{\nu^\alpha - 1} \) and \( C_{\nu^\alpha + \lambda} \), the edges incident to \( s \) and \( t \) in \( G^\alpha \) correspond to the edges of the two said mincuts, and every vertex in \( G^\alpha \) is reachable from \( s \) and reaches \( t \). Pick a path \( Q_1 \) from \( s \) to \( v \) in \( G^\alpha \) and let \( i_1 \) be the index of the path of the flow \( \mathcal{P} \) in the parent call \( \rho \). Let \( \rho \) contains the edge of \( C_{\nu^\alpha - 1} \) corresponding to the first edge of \( Q_1 \). Pick a path \( Q_2 \) from \( v \) to \( t \) in \( G^\alpha \) and let \( i_2 \) be the index of
the path of the flow \( \mathcal{P} \) in the parent call \( \rho \) that contains the edge of \( C_{b^2, +\lambda} \) corresponding to the last edge of \( Q_2 \). Then \((i_1, i_2)\) is the local signature of \( v \) at \( \rho' \). The signature of \( v \) is the sequence of all local signatures of \( v \), in the top-to-bottom order in the recursion tree. The signature of an edge \( e \) is the pair consisting of the signatures of its endpoints. Finally, we define \( \mathcal{B} \) as the partition of the deletable edges of \( \mathcal{P} \) according to the path of \( \mathcal{P} \) they belong to and according to their signatures. Since there are \( 2^{O(h_{\text{max}}(k) \log k)} \) signatures, \( |\mathcal{B}| \leq 2^{O(k^3 \log k)} \).

Let \( f_p(x) = 1 + 2px \), \( g_0(p) = 2p \), \( g_{i+1}(p) = f_p(g_i(p)) \) for \( 0 \leq i < h_{\text{max}}(k) \), and \( g(k, p) = h_{\text{max}}(k)(p) \); recall that \( h_{\text{max}}(k) = O(k^3) \) is the maximum possible depth of the recursion for a fixed value of \( k \). We have \( q(k, p) = 2^{O(k^3 \log (kp))} \). We prove, by the bottom-to-top induction over the recursion tree, that if the recursive call \( \rho \) at depth \( i \) applied to a tuple \((G, s, t, k, \mathcal{P}, \kappa)\) returned \((A, \mathcal{P})\), \( P \in \mathcal{P} \), and \( C \) and \( D \) are two disjoint sets of size \( q_{h-i}(p) \) consisting of deletable edges of \( E(P) \) of the same signature that are interlaced on \( P \), then \( G - \{s, t\} \) contains a family of \( p \) pairwise vertex-disjoint \( CD \)-soybeans. Here \( h \leq h_{\text{max}}(k) \) is the actual depth of the recursion.

In the subsequent paragraphs we consider different cases the algorithm of Theorem 7.1 can enter and in each of them prove the desired claim.

### 7.3 Initial steps

We first investigate the initial preprocessing steps.

These can be split into two types. The first type are leaves of the recursion: when \( \lambda_G(s, t) = 0 \) and when \( \lambda > k \). In the first case, the algorithm returns \( A = \emptyset \) and \( \mathcal{P} = \emptyset \), so there is nothing to prove. In the second case, the algorithm returns \( A = \{(s, t)\} \) and \( \mathcal{P} \) consisting of a single path \( P \) along the edge of \( A \), and again there is nothing to prove.

The second type of steps invoke one recursive call \( \rho' \) on a modified graph \( G' \), obtaining \((A', \mathcal{P}')\). Recall that in all cases, \( G' - \{s, t\} \) is a subgraph of \( G - \{s, t\} \). The algorithm returns \((A, \mathcal{P})\) that is constructed from \((A', \mathcal{P}')\) by setting \( A \) to be \( A' \) plus at most \( 2\lambda + 1 \leq 2k + 1 \) additional edges, all incident with \( s \) or \( t \), and \( \mathcal{P} \) to be \( \mathcal{P}' \) with possibly one additional one-edge path. The claim is again straightforward as \( q_{h-i-1}(k, p) \leq q_{h-i}(k, p) \) and the requested soybeans cannot use vertices \( s \) nor \( t \).

### 7.4 Base case

In the base case of the algorithm, \( \mathcal{P} \) is an \( st \)-maxflow and for every \( i \neq j \), \( i, j \in [\lambda] \), there is no path from \( V(P_i) \) to \( V(P_j) \) in \( G - \{s, t\} \). This is a place where we need to slightly modify the behavior of the algorithm of Theorem 7.1.

Let \( \mathcal{B} \) be the set of all bottleneck edges. For every \( i \in [\lambda] \), let \((u_{i, 1}, v_{i, 1}), \ldots, (u_{i, a_i}, v_{i, a_i})\) be the bottleneck edges on \( P_i \), in the order along \( P_i \). Denote \( v_{i, 0} = s \) and \( u_{i, a_i+1} = t \). For \( i \in [\lambda] \) and \( 0 \leq b \leq a_i \), let \( G_{i,b} \) be the subgraph of \( G \) induced by all vertices that are reachable from \( v_{i,b} \) and from where \( u_{i,b+1} \) is reachable in \( G - B \). Let \( G_i = \{(u_{i,b}, v_{i,b} \mid 1 \leq b \leq a_i \} \cup \bigcup_{b=0}^{a_i} G_{i,b} \). Note that the graphs \( G_i \) intersect only in vertices \( s \) and \( t \) and \( \lambda_G(s, t) = 1 \).

For a star \( st \)-cut \( Z \) of size at most \( k \), let \( Z_i = Z \cap E(G_i) \). The analysis of the base case shows that \( Z_i \) is a star \( st \)-cut in \( G_i \) and \( \text{core}_{G_i}(Z) = \bigcup_{i \in [\lambda]} \text{core}_{G_i}(Z_i) \). Furthermore, either \( \text{core}_{G_i}(Z_i) \) consists of a single bottleneck edge and \( Z_i \) contains no other bottleneck edge on \( P_i \), or \( Z_i \) does not contain any bottleneck edge of \( P_i \) at all.

For every \( i \in [\lambda] \), we proceed as follows. We guess integers \( 1 \leq \kappa_i \leq k \) such that \( \sum_{i \in [\lambda]} \kappa_i \geq \kappa \) and \( \sum_{i \in [\lambda]} k_i \leq k \). We aim at \( k_i = |Z_i| \) and \( \kappa_i = |\text{core}_{G_i}(Z_i)| \). This happens in total with probability \( 2^{-O(k \log k)} \).

If \( \kappa_i = 1 \), we aim at capturing star \( st \)-cuts \( Z_i \) with \( \text{core}_{G_i}(Z_i) \) consisting of a single bottleneck edge. We set \( A_i = \{(u_{i,b}, u_{i,b+1}) \mid 0 \leq b \leq a_i \} \) and \( P_i' \) to be a path consisting of edges
(s, u_i, 1), (u_i, 1, v_i, 1), (v_i, 1, u_i, 2), \ldots, (u_{i,g_i}, v_{i,a_i}, (v_{i,a_i}, t)). The algorithm returns \( A_i \) as part of the set \( A \) and \( P_i' \) as one of the flow paths in \( \mathcal{P} \).

For our desired claim, observe that the deletable edges on \( P_i' \) are only bottleneck edges. Thus, given \( C \) and \( D \) interlaced on \( P_i' \), each of size at least \( q_0(k, p) = 2p \), one can construct the desired soybeans as follows: If \( c_1, c_2, \ldots, c_{2p} \) and \( d_1, d_2, \ldots, d_{2p} \) are the first \( 2p \) edges of \( C \) and \( D \), respectively, and \( P_{i,j} \) is the subpath of \( P_i \) from \( c_{2j} \) to \( d_{2j} \), then \( \{P_{i,j} \mid j \in [p]\} \) is the desired family of soybeans. So we can put the whole \( A_i \) as a single set in \( \mathcal{B} \).

If \( \kappa_i > 1 \), we aim at capturing star \( st \)-cuts \( Z_i \) that do not contain any bottleneck edge on \( P_i \). Let \( A_{i,b} \) be the set of copies of all bottleneck edges on \( P_i \). For every \( 0 \leq b \leq a_i \), we recurse on graphs \( G_{i,b} \) with \( v_{i,b} \) playing the role of \( s \) and \( u_{i,b+1} \) playing the role of \( t \), parameters \( k_i \), \( \kappa_i \), and a flow consisting of a single flow path \( P_i \) from \( v_{i,b} \) to \( u_{i,b+1} \). Let \( (A_{i,b}, \mathcal{P}_{i,b}) \) be the returned pair. The returned set \( A \) consists of, for every \( i \in [\lambda] \), the set \( A_i = A_i^1 \cup \bigcup_{b=0}^{a_i} A_{i,b} \). The returned set \( \mathcal{P} \) consists of, for every \( i \in [\lambda] \), \( \kappa_i \) flow paths, combined from flow paths \( \mathcal{P}_{i,b} \) (recall that each \( \mathcal{P}_{i,b} \) is of size at least \( \kappa_i \) \( \kappa_i \)) concatenated using edges of \( A_{i,b} \).

For our desired claim, consider a returned path \( P \in \mathcal{P} \) and interlaced sets \( C, D \) of \( q_{h-i}(k, p) \) deletable edges on \( P \). We have two cases. First, there exists an integer \( 0 \leq b \leq a_i \) such that \( G_{i,b} \) contains at least \( q_{h-i-1}(k, p) + 1 \) edges of \( C \). Then, the path of \( \mathcal{P}_{i,b} \) used to construct \( P \) contains interlaced subsets of \( C, D \) of size \( q_{h-i-1}(k, p) \). The claim follows from the inductive hypothesis for the recursive call on \( G_{i,b} \).

In the second case, there are at least \( 2p + 1 \) indices \( b \) such that \( G_{i,b} \) contains an edge of \( C \). Consequently, there are indices \( b_1 \leq b_1' < b_2 \leq b_2' < \ldots < b_p \leq b_p' \) such that for every \( j \in [p] \), \( G_{i,b_j} \) contains an edge of \( C \) and \( G_{i,b_j'} \) contains an edge of \( D \). For \( j \in [p] \), let \( Q_j \) be a path from \( v_{j,b_j} \) to \( u_{j,b_j+1} \) containing an edge of \( C \) from \( G_{i,b_j} \) and let \( Q_j' \) be a path from \( v_{j,b_j} \) to \( u_{j,b_j+1} \) containing an edge of \( D \) from \( G_{i,b_j} \). Such paths exist by the construction of the graphs \( G_{i,b} \). Then, \( \{Q_j Q_j' \mid j \in [p]\} \) is the desired soybean harvest.

### 7.5 Small \( \ell \) case

In the small \( \ell \) case the situation is very similar to the second type of initial steps.

The algorithm always invokes one recursive call, on a graph \( G' \) such that \( G' - \{s, t\} \) is a subgraph of \( G - \{s, t\} \), obtaining \( (A', \mathcal{P}') \). The returned set \( A \) consists of \( A' \) and additional edges \( A_0 \) with \( |A_0| \leq 4\lambda b^3 \log b + 2 \leq 16k^3 + 14 \), all incident with \( s \) and \( t \). The returned flow \( \mathcal{P} \) consists of the paths \( \mathcal{P}' \), possibly with an edge of \( A_0 \) added at the beginning or end, and possibly one additional path that contains at most one deletable edge.

Thus, the claim follows directly from the inductive hypothesis for the recursive subcall.

### 7.6 Large \( \ell \) case

In the large \( \ell \) case the situation is quite similar to the base case, but a bit more complex. Let \( (G, s, t, k, \mathcal{P}, \kappa) \) be the input to the recursive call in question.

The algorithm recurses on graphs \( G^\alpha \) for all excellent indices \( \alpha \), obtaining pairs \( (A^\alpha, \mathcal{P}^\alpha) \). Observe that in the returned flow \( \mathcal{P} \), the only deletable edges are those in graphs \( G^\alpha \) on paths \( \mathcal{P}^\alpha \). Fix \( P \in \mathcal{P} \); the path \( P \) consists of edges of \( A \) and some flow paths from flows \( \mathcal{P}^\alpha \). Assume that we have interlaced sets \( C, D \) of deletable edges on \( P \) of the same signature, each of size \( q_{h-i}(k, p) \).

As in the base case, there are two cases. First, there exists \( \alpha \) such that \( C \) contains at least \( q_{h-i-1}(k, p) + 1 \) edges in \( G^\alpha \). Then, the flowpath \( P' \in \mathcal{P}^\alpha \) contained in \( P \) contains interlaced subsets \( C', D' \) of \( C \) and \( D \) of size \( q_{h-i-1}(k, p) \) each. The claim follows from the inductive hypothesis.
In the second case, there are at least $2p + 1$ indices $\alpha$ for which there is an edge of $C$ in $G^\alpha$. Thus, there are indices $\alpha_1 \leq \alpha_1' < \alpha_2 \leq \alpha_2' < \alpha_3 \leq \ldots < \alpha_p < \alpha_p'$ such that for every $j \in [p]$, there is an edge $c_j$ of $C$ in $G^{\alpha_1'}$ and an edge $d_j$ of $D$ in $G^{\alpha_p'}$.

Recall that all edges of $C$ and $D$ are of the same signature. That is, there are two paths $P^1, P^2 \in \mathcal{P}$ such that, for edge $e \in C \cup D$, if $e$ lies in $G^\alpha$ then there is a walk $W(e)$ in $G$ from the head of the edge of $E(P^1) \cap C_{b_i \lambda} - 1$ to the tail of the edge of $E(P^2) \cap C_{b_j \lambda}\lambda$ that contains $e$ and is completely contained between $C_{b_i \lambda} - 1$ and $C_{b_j \lambda}\lambda$. For every $j \in [p]$, let $W_j$ be a concatenation of $W(c_j)$ and a subpath of $P^2$ from the tail of the edge of $E(P^2) \cap C_{\alpha_p' + \lambda}$ to the tail of the edge of $E(P^2) \cap C_{\alpha_p' + \lambda}$ and let $W'_j$ be a concatenation of and a subpath of $P^1$ from the head of the edge of $E(P^1) \cap C_{\alpha_j' \lambda}$ to the head of the edge of $E(P^1) \cap C_{\alpha_j' \lambda} - 1$ and $W(d_j)$. (Recall that $\alpha_j \leq \alpha_j'$.) Then, $\{W_j W'_j \mid j \in [p]\}$ is the desired soybean family.

8 Two-terminal-pair Weighted Directed Multicut is $W[1]$-hard

In the Weighted Directed Multicut problem, the input is a directed graph $G$, a set of terminal pairs $\{(s, t_i) : i \in [p]\}$, a weight function on the vertex set $w : V(G) \to \mathbb{N}$ and positive integers $k$ and $W$. For a subset $S \subseteq V(G)$ we define $w_S(G) := \sum_{v \in S} w(v)$. The goal is to determine whether there exists a subset $S \subseteq V(G)$ such that $|S| \leq k$, $w(S) \leq W$ and $G - S$ has no $s,t_i$-path for each $i \in [p]$. In this section, we show that Weighted Directed Multicut is $W[1]$-hard parameterized by $k$, even with two terminal-pairs (that is when $p = 2$). In fact, we show that it does not admit an $f(k) \cdot n^{o(k/\log k)}$ algorithm under the ETH. We denote this problem with two terminal pairs by 2-Wt-DMC. The hardness proof we provide is essentially a simplification of the reduction given by Pilipczuk and Wahlström [25] for proving the W[1]-hardness of the Directed Multicut problem with four terminal-pairs. Our reduction essentially demonstrates that the synchronization of some gadgets achieved in the reduction in [25] using two additional terminal-pairs can also be achieved if the vertices are allowed polynomial (in the input size) weights. This helps us to eliminate two terminal-pairs in the reduction of [25] at the cost of adding polynomial weights.

**Theorem 8.1.** Weighted Directed Multicut is $W[1]$-hard even for two terminal pairs. Furthermore, assuming the ETH, the problem cannot be solved in $f(k) \cdot n^{o(k/\log k)}$ time, where $n$ is the number of vertices of the input graph.

To prove Theorem 8.1, we give a reduction from Partitioned Subgraph Isomorphism (PSI), parameterized by the number of edges in the pattern graph. In the Partitioned Subgraph Isomorphism problem, given two undirected graphs $G,H$ such that $V(G) = \bigcup_{i \in V(H)} V_i$, the goal is to determine if there exists a homomorphism $\xi : V(H) \to V(G)$ such that $\xi(i) \in V_i$ for each $i \in V(H)$. This problem has been shown to be $W[1]$-hard parameterized by $|E(H)|$ by Marx [22, Corollary 6.3]. In fact, the authors show that there is no $f(k) \cdot n^{o(k/\log k)}$ algorithm for PSI, where $k = |E(H)|$ and $n$ is the number of vertices in the input graph, unless the ETH fails.

**Proof of Theorem 8.1.** Let $(G,H)$ be an instance of PSI where $V(G) = \bigcup_{i \in V(H)} V_i$. Without loss of generality, let $V(H) = \{1, \ldots, h\}$, let $|V^i| = |V^j| = n$ for each $i, j \in V(H)$ and assume that $H$ has no isolated vertices. Let $V^i = \{v^i_1, \ldots, v^i_n\}$. Let $k = |E(H)|$, then $|V(H)| \leq 2k$ since $H$ has no isolated vertices. We now construct an instance $(D_i, \{s_i, t_i\}_{i \in [2]}, w, k', W)$ of 2-Wt-DMC. Set $k' = 5k + h$ and $W = M(2k(n + 1) + h) + k$, where $M = k + 1$. 

27
Figure 4: An illustration of the construction of the paths $X^{i,j}$, $Y^{i,j}$, $X^{j,i}$, $Y^{j,i}$ and the grid $P^{i,j}$, where $i < j$ and $n = 5$. The brown vertices whose labels have subscript $a$ have weight $Ma$. The green vertices whose labels have subscript $a$ have weight $M(n + 1 - a)$. The black vertices have weight 1 and the white vertices are undeletable. The red, and the blue, highlighted paths are the $s_1t_1$-path, and $s_2t_2$-path respectively, that survive after deleting the solution vertices from $X^{i,j}$, $Y^{i,j}$, $X^{j,i}$ and $Y^{j,i}$. The unique common intersection point of the two highlighted paths is $p_{a,b}$, which is picked by the solution. The vertices picked by the solution are encircled in red.
Construction of $D$: For each $i \in V(H)$, let $Z_i$ be the path $(z_{i,n}^i, \hat{z}_{i,n}^i, z_{i,n-1}^i, \ldots, z_{i,1}^i, z_{i,0}^i)$ on $2n+1$ vertices. These paths are called the $Z$-paths and are added to $D$. For each ordered pair $(i, j) \in V(H) \times V(H)$, such that $\{i,j\} \in E(H)$, let $X_{i,j} = (x_{i,n}^{i,j}, \hat{x}_{i,n}^{i,j}, x_{i,n-1}^{i,j}, \ldots, x_{i,1}^{i,j}, \hat{x}_{i,1}^{i,j}, x_{i,0}^{i,j})$ and $Y_{i,j} = (y_{i,n}^{i,j}, \hat{y}_{i,n}^{i,j}, y_{i,n-1}^{i,j}, \ldots, y_{i,1}^{i,j}, \hat{y}_{i,1}^{i,j}, y_{i,0}^{i,j})$ be paths on $2n+1$ vertices each. These paths are called $X$-paths and $Y$-paths, respectively, and are also added to $D$. For each $a \in \{0, \ldots, n\}$, set $\text{wt}(z_{i,a}^i) = \text{wt}(\hat{x}_{i,a}^{i,j}) = \text{wt}(\hat{y}_{i,a}^{i,j}) = W + 1$, that is, these vertices are undeletable. Further set $\text{wt}(z_{a}^i) = M$, $\text{wt}(\hat{x}_{a}^{i,j}) = Ma$ and $\text{wt}(\hat{y}_{a}^{i,j}) = M(n+1-a)$. Observe that $\text{wt}(\hat{x}_{a}^{i,j}) + \text{wt}(\hat{y}_{a}^{i,j}) = M(n+1)$.

Next, we describe the gadget that synchronizes the vertices that a solution for $2$-$\text{WtDMC}$ picks from the paths described above. For each ordered pair $(i,j) \in V(H) \times V(H)$ such that $\{i,j\} \in E(H)$, and for every $a \in \{0,1, \ldots, n\}$, we add the edges $(x_{i,a}^{i,j}, z_{i,a}^i)$ and $(z_{i,a}^i, y_{a}^{i,j})$ to $D$ (see Figure 5 for an illustration). Further we add two terminal-pairs $(s_1, t_1)$ and $(s_2, t_2)$ together with the following incident edges. For each $(i,j) \in E(H)$, such that $i < j$, the edges $(s_1, x_{i,j}^{i,j}), (x_{i,j}^{i,j}, t_1), (s_1, y_{i,j}^{i,j}), (y_{i,j}^{i,j}, t_1)$ are added to $D$. Further the edges $(s_2, x_{n,j}^{i,j}), (x_{0,j}^{i,j}, t_2), (s_2, y_{n,j}^{i,j}), (y_{0,j}^{i,j}, t_2)$ are added to $D$. The edges described here ensure that for each $(i,j) \in V(H) \times V(H)$ such that $(i,j) \in E(H)$, there exists an $a \in \{0,1, \ldots, n\}$ such that the solution for $2$-$\text{WtDMC}$ picks $z_{i,a}^i, x_{a,j}^{i,j}$, and $y_{a,j}^{i,j}$.

The next gadget ensures that vertices of the $2$-$\text{WtDMC}$ solution in the paths described above are the map of a valid homomorphism. For every edge $(i,j) \in E(H)$ such that $i < j$, we add a grid $P_{i,j}$ on the vertex set $\{p_{a,b}^{i,j} \mid a,b \in \{1, \ldots, n\}\}$ to $D$. The edge set contains the column edges $\{(p_{a,b}^{i,j}, p_{a+1,b}^{i,j}) \mid a \in \{1, \ldots, n-1\}, b \in \{1, \ldots, n\}\}$, and the row edges $\{(p_{a,b}^{i,j}, p_{a,b+1}^{i,j}) \mid a \in \{1, \ldots, n\}, b \in \{1, \ldots, n-1\}\}$. The weight function of the vertices of this grid is defined as follows: $\text{wt}(p_{a,b}^{i,j}) = 1$ if $(v_i^a, v_i^b) \in E(G)$, and $\text{wt}(p_{a,b}^{i,j}) = W$ otherwise.

Further we add edges to $D$ connecting the paths $X_{i,j}^{i,j}, Y_{i,j}^{i,j}, X_{i,j}^{j,i}$ and $Y_{i,j}^{j,i}$ to $P_{i,j}^{i,j}$. For every $a \in \{1, \ldots, n\}$, the following edges are added: $(x_{i,a}^{i,j}, p_{a,j}^{i,j}), (p_{a,n}^{i,j}, y_{a-1}^{i,j}), (x_{i,j}^{j,i}, p_{n,a}^{j,i})$, and $(p_{0,j}^{j,i}, y_{0,j}^{j,i})$. This finishes the construction of $D$. See Figure 4 for an illustration of the construction.

We next show that $(G,H)$ is a $\text{Yes}$-instance of $\text{PSI}$ if and only if $(D, k', (s_1, t_1)_{i \in [2]}, \text{wt}, W)$ is a $\text{Yes}$-instance of $2$-$\text{WtDMC}$.

**Forward direction.** Let $\phi : V(H) \rightarrow \{1, \ldots, n\}$ be a solution to the $\text{PSI}$-instance $(G,H)$, that is, for every $(i,j) \in E(H)$, we have $(v_i^\phi(i), v_j^\phi(j)) \in E(G)$. We choose $S := \{(i,j) \in V(H) \times V(H) \mid \phi(i) = \phi(j) \}$ and claim that it is a solution to the constructed $2$-$\text{WtDMC}$-instance $(D, k', (s_1, t_1)_{i \in [2]}, \text{wt}, W)$. Observe that the set $S$ contains a vertex from $Z_i^i, X_{i,j}^{i,j}, Y_{i,j}^{i,j}$ and the grid $P_{i,j}^{i,j}$ for every $(i,j) \in V(H) \times V(H)$ with $(i,j) \in E(H)$. Further observe that
\(|S| = 5k + h\) and \(\text{wt}(S) = M(2k(n + 1) + h) + k\) since the weight of the vertices on the \(X\)- and \(Y\)-paths in \(S\) is \(2kM(n + 1)\), the weight of the vertices of \(Z\)-paths in \(S\) is \(Mh\) and the weight of the grid vertices in \(S\) is \(k\).

We claim that \(D - S\) has no \(s_1t_1\)-path and no \(s_2t_2\)-path. Here, we show that \(D - S\) has no \(s_1t_1\)-path. That there is also no \(s_2t_2\)-path follows by symmetric arguments. Observe from the construction that every \((s_1,t_1)\) path contains \(x^n_i\) or \(y^n_j\), for some \(i, j\), as its first internal vertex. There are four kinds of \(s_1t_1\)-paths in \(D\). The first one traverses \(X^{i,j}\) fully until \(x^n_0\) (which is an in-neighbour of \(t_1\)). The second one traverses \(Y^{i,j}\) fully. Both, paths of the first and the second kind, are hit by \(S\) as \(S\) contains a vertex from \(X^{i,j}\) and a vertex from \(Y^{i,j}\).

The third kind of \(s_1t_1\)-path traverses some subpath of \(X^{i,j}\), say until the vertex \(x^n_0\), jumps to \(Z^i\) at vertex \(z^n_i\), traverses \(Z^i\) until say \(z^b_i\) where \(b \leq a\), jumps to \(Y^{i,j}\) at vertex \(y^n_j\), and then traverses \(Y^{i,j}\) until \(y^n_0\). Since \(\hat{x}^{i,j}_{\phi(i)}, \hat{y}^{i,j}_{\phi(i)}, z^n_i \in S\), if such a path exists then \(a \geq \phi(i)\) and \(b \leq \phi(i)\), thus the \(y^n_jy^n_0\)-subpath of \(Y^{i,j}\) traversed by such an \(s_1, t_1\)-path contains a vertex of \(S\), namely \(y^n_j\).

The last kind of \(s_1, t_1\)-path passes through the grid \(P_{i,j}\). Such a path traverses \(X^{i,j}\) until \(x^n_0\), jumps to the first column of the grid at the vertex \(p^n_{i,1}\), traverses the grid \(P^{i,j}\) to get to a vertex \(p^n_{i,n}\) (there are potentially many ways to reach this vertex inside the grid), for some \(b \geq a\), then jumps to \(Y^{i,j}\) at the vertex \(y^n_j\), and then traverses \(Y^{i,j}\) until \(y^n_0\). Since \(\hat{x}^{i,j}_{\phi(i)} \in S\), if \(a < \phi(i)\), the path is hit by \(S\). Since \(\hat{y}^{i,j}_{\phi(i)} \in S\), if \(\phi(i) < b\), the path is hit by \(S\). Otherwise \(a = b = \phi(i)\). In this case the \(s_1t_1\)-path under consideration uses exactly the vertices of the \(p^{i,j}_{\phi(i)}, p^{i,j}_{\phi(i),n}\)-subpath of the grid. Since \(p^{i,j}_{\phi(i),\phi(j)} \in S\), this kind of \(s_1t_1\)-path is again hit by \(S\).

Thus, \(S\) is a solution of the 2-Wt-DMC-instance \((D, k', (s_i, t_i)) \in \mathcal{E}[2], \text{wt}, W\).

**Reverse direction.** Let \(S \subseteq V(D)\) be a solution for the 2-Wt-DMC-instance \((D, k', (s_i, t_i)) \in \mathcal{E}[2], \text{wt}, W\). We construct a function \(\phi : V(H) \to \{1, \ldots, n\}\) such that for each \((i, j)\) \(\in E(G), (u^{i,j}_{\phi(i)}, v^{j}_{\phi(j)}) \in E(G)\).

Note that, since for every path \(X^{i,j}\) and for every path \(Y^{i,j}\), there is an \(s_1t_1\)-path or an \(s_2t_2\)-path containing only this path, \(S\) contains at least one vertex from each \(X^{i,j}\) and each \(Y^{i,j}\). Also, since the vertices \(x^n_i\) and \(y^n_j\) are undeletable, the set \(S\) contains a deletable vertex of \(Z^i\), as otherwise there is an \(s_1t_1\)-path or an \(s_2t_2\)-path starting from \(x^n_i\) and then jumping to the path \(Z^i\) at the vertex \(z^n_i\), traversing the path \(Z^i\) until \(z^n_b\), then jumping to \(y^n_0\) (which is an in-neighbour of \(t_1\)).

Fix \((i, j) \in V(H) \times V(H)\) such that \((i, j) \in E(H)\) and \(i < j\). The case when \(j > i\) is symmetric. We eventually show that \(S\) intersects \(X^{i,j}, Y^{i,j}\) and \(Z^i\), each at exactly one vertex. Towards this, let \(a \in \{1, \ldots, n\}\) be the smallest index such that \(\hat{x}^{i,j}_a \in S\), and let \(b \in \{1, \ldots, n\}\) be the smallest index such that \(\hat{y}^{i,j}_b \in S\). We first claim that \(a \geq b\). Suppose \(a < b\), then consider the following \(s_1t_1\)-path in \(G - S\). The path first visits \(x^n_i\), traverses \(X^{i,j}\) until reaching \(x^n_{a}\) (note that none of the vertices on this subpath belong to \(S\) so far, either due to the choice of \(a\) or because they are undeletable), jumps to \(z^n_a\) (note that this vertex is undeletable), and then jumps to \(y^n_j\) and finally traverses \(Y^{i,j}\) until reaching \(y^n_0\) (again, none of the vertices of this subpath belongs to \(S\) either due to the choice of \(b\) or because they are undeletable). This is a contradiction to the fact that \(S\) is a solution.

From the above paragraph, \(\text{wt}(S) \cap (X^{i,j} \cup Y^{i,j}) \geq M \cdot a + M \cdot (n + 1 - b) \geq M \cdot (n + 1)\). Note that if \(a > b\), then \(\text{wt}(S) \cap (X^{i,j} \cup Y^{i,j}) > M \cdot (n + 1) + M\). Thus, if \(S\) picks exactly one vertex from each \(X\)-, \(Y\)- and \(Z\)-paths, then the weight of \(S\) restricted to the vertices in these paths is \(4kM \cdot (n + 1) + Mh\). Since the total weight of \(S\) is \(4kM \cdot (n + 1) + Mh + k\), and the weight of each vertex on the paths \(X^{i,j}, Y^{i,j}\) or \(Z^i\) is at least \(M = k + 1\), we conclude that \(S\) indeed picks
exactly one vertex from each of the $X$-, $Y$- and $Z$-paths. Moreover, $a = b$. That is, $S \cap X^{i,j} = \hat{x}_a^{i,j}$ and $S \cap Y^{i,j} = \hat{y}_a^{i,j}$.

We now show that $S \cap Z^i = \hat{z}_a^i$. Suppose not, then consider the following $s_1 t_1$-path in $G - S$. It first visits $x_{\phi(i)}^{i,j}$, traverses $X^{i,j}$ until reaching $x_{\phi(i)}^{i,j}$, jumps to $Z^i$ at the vertex $z_{\phi(i)}^i$, traverses $Z^i$ until reaching $z_{\phi(i)}^{i,j}$ (note that the only deletable vertex on this subpath is $z_{\phi(i)}^i$, which by our assumption is not in $S$), jumps to the path $Y^{i,j}$ at the vertex $y_{a-1}^{i,j}$, and then traverses $Y^{i,j}$ until reaching $y_a^{i,j}$. Again, this yields a contradiction to $S$ being a solution.

So far, we have concluded that $S$ intersects each of the paths $X^{i,j}$, $Y^{i,j}$ and $Z^i$ in exactly one vertex. In fact, there exists $a \in \{1, \ldots, n\}$ such that $S \cap X^{i,j} = \hat{x}_a^{i,j}$, $S \cap Y^{i,j} = \hat{y}_a^{i,j}$ and $S \cap Z^i = \hat{z}_a^i$. Hence, to construct $\phi$, define $\phi(i) = a$.

Fix $(i, j) \in V(H) \times V(H)$ and $i < j$. Consider the $s_1 t_1$-path, we call it $P_1$, that first visits $x_{\phi(i)}^{i,j}$, then traverses $X^{i,j}$ until $x_{\phi(i)}^{i,j}$, jumps to the first column of the grid $P^{i,j}$ at vertex $p_{\phi(i),1}^{i,j}$, traverses the path in the $\phi(i)$-th row of $P^{i,j}$, that is the subpath from $(p_{\phi(i),1}^{i,j}$ to $p_{\phi(i),n}^{i,j}$), jumps to the path $Y^{i,j}$ at the vertex $y_{\phi(i)}^{i,j}$ and then traverses $Y^{i,j}$ until reaching $y_a^{i,j}$. Similarly, consider the $s_2 t_2$-path, we call it $P_2$, that first visits $x_{\phi(j)}^{j,i}$, traverses the path $X^{j,i}$ until reaching $x_{\phi(j)}^{j,i}$, jumps to the first row of the grid $P^{j,i}$ at the vertex $p_{1,\phi(j)}^{j,i}$ traverses the path in the $\phi(j)$-th column of $P^{j,i}$, that is the subpath from $(p_{1,\phi(j)}^{j,i}$ to $p_{n,\phi(j)}^{j,i}$), jumps to the path $Y^{j,i}$ at $y_{\phi(j)}^{j,i}$ and then traverses $Y^{j,i}$ until reaching $y_a^{j,i}$. Observe that the paths $P_1$ and $P_2$ do not intersect any of the vertices of $S$ that are on some $X$-, $Y$- or $Z$-path. Also, the paths $P_1$ and $P_2$ intersect at exactly one vertex of the grid $P_{i,j}$, which is $p_{\phi(i),\phi(j)}^{i,j}$. Since $S$ has at most $k$ vertices that are in none of the $X$-, $Y$- or $Z$-path, $S$ contains at most one vertex from each $P_{i,j}$. Thus, $p_{\phi(i),\phi(j)}^{i,j} \in S$.

As $\text{wt}(S) \leq W$, we can deduce from the construction of $D$ that $(v_{\phi(i)}^{i,j}, v_{\phi(j)}^{j,i}) \in E(G)$.

\[ \square \]

References

[1] Édouard Bonnet, Dibyayan Chakraborty, Eun Jung Kim, Nolene Köhler, Raul Lopes, and Stéphan Thomassé. Twin-width VIII: delineation and win-wins. CoRR, abs/2204.00722, 2022. arXiv:2204.00722, doi:10.48550/arXiv.2204.00722.

[2] Édouard Bonnet, Ugo Giocanti, Patrice Ossona de Mendez, Pierre Simon, Stéphan Thomassé, and Szymon Torunczyk. Twin-width IV: ordered graphs and matrices. In Stefano Leonardi and Anupam Gupta, editors, Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing (STOC 2022), pages 924–937. ACM, 2022. doi:10.1145/3519935.3520037.

[3] Édouard Bonnet, Eun Jung Kim, Stéphan Thomassé, and Rémi Watrigant. Twin-width I: tractable FO model checking. Journal of the ACM, 69(1):3:1–3:46, 2022. doi:10.1145/3486655.

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

[5] Jianer Chen, Yang Liu, Songjian Lu, Barry O’Sullivan, and Igor Razgon. A fixed-parameter algorithm for the directed feedback vertex set problem. J. ACM, 55(5), 2008.

[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] Rajesh Chitnis, László Egri, and Dániel Marx. List $H$-coloring a graph by removing few vertices. *Algorithmica*, 78(1):110–146, 2017. doi:10.1007/s00453-016-0139-6.

[8] Rajesh Hemant Chitnis, Marek Cygan, Mohammad Taghi Hajiaghayi, and Dániel Marx. Directed subset feedback vertex set is fixed-parameter tractable. *ACM Trans. Algorithms*, 11(4):28:1–28:28, 2015. doi:10.1145/2700209.

[9] Rajesh Hemant Chitnis, MohammadTaghi Hajiaghayi, and Dániel Marx. Fixed-parameter tractability of directed multiway cut parameterized by the size of the cutset. *SIAM J. Comput.*, 42(4):1674–1696, 2013.

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

[11] Marek Cygan, Pawel Komosa, Daniel Lokshtanov, Marcin Pilipczuk, Michal Pilipczuk, Saket Saurabh, and Magnus Wahlström. Randomized contractions meet lean decompositions. *ACM Trans. Algorithms*, 17(1):6:1–6:30, 2021. doi:10.1145/3426738.

[12] Marek Cygan, Marcin Pilipczuk, Michal Pilipczuk, and Jakub Onufry Wojtaszczyk. On multiway cut parameterized above lower bounds. *TOCT*, 5(1):3, 2013. URL: http://doi.acm.org/10.1145/2462896.2462899, doi:10.1145/2462896.2462899.

[13] Sylvain Guillemot. FPT algorithms for path-transversal and cycle-transversal problems. *Discrete Optimization*, 8(1):61–71, 2011.

[14] Yoichi Iwata, Magnus Wahlström, and Yuichi Yoshida. Half-integrality, lp-branching, and FPT algorithms. *SIAM J. Comput.*, 45(4):1377–1411, 2016. doi:10.1137/140962838.

[15] Eun Jung Kim, Stefan Kratsch, Marcin Pilipczuk, and Magnus Wahlström. Directed flow-augmentation. *CoRR*, abs/2111.03450, 2021. URL: https://arxiv.org/abs/2111.03450, arXiv:2111.03450.

[16] Stefan Kratsch, Tomáš Masařík, Irene Muzi, Marcin Pilipczuk, and Manuel Sorge. Optimal discretization is fixed-parameter tractable. In Dániel Marx, editor, *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms, SODA 2021, Virtual Conference, January 10 - 13, 2021*, pages 1702–1719. SIAM, 2021. doi:10.1137/1.9781611976465.103.

[17] Stefan Kratsch, Marcin Pilipczuk, Michal Pilipczuk, and Magnus Wahlström. Fixed-parameter tractability of multicut in directed acyclic graphs. *SIAM J. Discrete Math.*, 29(1):122–144, 2015. URL: http://dx.doi.org/10.1137/120904202, doi:10.1137/120904202.

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

[19] 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.

[20] Adam Marcus and Gábor Tardos. Excluded permutation matrices and the stanley-wilf conjecture. *J. Comb. Theory, Ser. A*, 107(1):153–160, 2004. doi:10.1016/j.jcta.2004.04.002.
A Hardness of arbitrary CSPs with permutation constraints

In this section we explain in bigger detail the claim from the introduction that without any control on the complexity of permutation constraints, the obtained CSP instance is W[1]-hard when parameterized by the number of variables and permutation constraints.

Recall the classic W[1]-hard MULTICOLORED CLIQUE problem: the input consists of an integer $k$ being the parameter, a graph $G$, and a partition of $V(G)$ into $k$ independent sets $V_1, V_2, \ldots, V_k$; the goal is to find a $k$-clique in $G$, which necessarily needs to contain exactly one vertex from each set $V_i$. By a padding argument, we can assume $|V_1| = |V_2| = \ldots = |V_k| = n$.

For each $1 \leq i \leq k$, enumerate the vertices of $V_i$ as $v_{i,0}, v_{i,1}, \ldots, v_{i,n-1}$.

For every $1 \leq i \leq k$, create a variable $x_i$ with domain $\{0, 1, \ldots, n-1\}$; setting the value $x_i = a$ corresponds to choosing a vertex $v_{i,a}$ to our clique. For every $1 \leq i, j \leq k$, $i \neq j$, create a variable $y_{i,j}$ with domain $\{0, 1, \ldots, n-1\} \times \{0, 1, \ldots, n-1\}$ ordered lexicographically; setting the value $y_{i,j} = (a, b)$ corresponds to choosing vertices $v_{i,a}$ and $v_{j,b}$ to our clique. Bind the variables using the following constraints:

- For every $1 \leq i, j \leq k$, $i \neq j$, we express a constraint
  \[ \bigwedge_{0 \leq a, b < n} y_{i,j} = (a, b) \implies x_i = a \]
  as the following conjunction:
  \[ \left( \bigwedge_{0 \leq a < n-1} (x_i \leq a) \lor (y_{i,j} \geq (a + 1, 0)) \right) \land \left( \bigwedge_{1 \leq a < n} (y_{i,j} \leq (a - 1, n - 1)) \lor (x_i \geq a) \right). \]

- For every $1 \leq i < j \leq k$, we introduce a permutation constraint
  \[ y_{j,i} = \pi(y_{i,j}), \]
  where $\pi : \{0, 1, \ldots, n-1\} \times \{0, 1, \ldots, n-1\} \to \{0, 1, \ldots, n-1\} \times \{0, 1, \ldots, n-1\}$ is defined as $\pi(a, b) = (b, a)$.

Finally, for every $1 \leq i < j \leq k$, we restrict the permutation constraint between $y_{i,j}$ and $y_{j,i}$ to only those values $y_{i,j} = (a, b)$ where $v_{i,a}v_{j,b} \in E(G)$.

The above is an encoding of the input MULTICOLORED CLIQUE instance a CSP instance with $k + k(k-1) = k^2$ variables and $\binom{k}{2}$ permutation constraints. Hence, we cannot hope for an FPT algorithm for our CSP instances, with only the number of variables and permutation constraints as parameters; we need some structural parameter of the obtained permutation constraints. Note that the permutation $\pi$ used above has a grid minor of size $n$ in its permutation matrix.

33
B Twin-width to grid-rank

Theorem B.1 (Marcus and Tardos [20]). For every integer \( k \), there is some \( c_k \leq 2^{O(k \log k)} \) such that every \( n \times n \) 0-1 matrix \( A \) with at least \( c_k \max(n,m) \) 1-entries has a \( k \)-grid minor. Moreover, if it exists, such a grid minor can be found in \( 2^{O(k \log k)} n^{O(1)} \) time.

Proposition B.2. Let \( n \geq 2 \) and \( a_1, \ldots, a_n \) be non-negative integers, and let \( s = \sum_{i=1}^n a_i/n \). Then, there exists an \( i \in [n-1] \) such that \( a_i + a_{i+1} \leq 4s - 1 \).

Proof. Suppose for a contradiction that for all \( i \in [n-1] \), \( a_i + a_{i+1} > 4s \). First suppose that \( n \) is even. Then,
\[
\sum_{i=1}^n a_i = a_1 + a_2 + \sum_{i=3}^n a_i \geq 4s + \sum_{i=3}^n a_i \geq \cdots \geq \frac{n}{2}4s > ns,
\]
a contradiction. Next suppose that \( n \) is odd which means that \( n \geq 3 \). In this case,
\[
\sum_{i=1}^n a_i = a_1 + \sum_{i=2}^n a_i \geq a_1 + \frac{n-1}{2}4s \geq 2(n-1)s = ns \left( 2 - \frac{2}{n} \right) > ns,
\]
a contradiction since \( 2/n < 1 \). \( \square \)

Proposition B.3. Let \( A \) be a 0-1 matrix, let \( k \) be a positive integer, and let \( c_k \) be the constant from Theorem B.1. One can in \( 2^{O(k \log k)} n^{O(1)} \) time find either
- a \( k \)-grid minor in \( A \), or
- a \( 4c_k \)-contraction sequence of \( A \), respecting the order of the rows and columns of \( A \).

Proof. We do the following greedily. Assume the number of rows in \( A \) is at least the number of its columns, otherwise we swap the roles of rows and columns. Let \( o \) be the average number of 1-entries in each row of \( A \). If \( o > c_k \), then by Theorem B.1, we can find a \( k \)-grid minor of \( A \) in \( 2^{O(k \log k)} n^{O(1)} \) time and we are done. Otherwise, \( o \leq c_k \), which implies by Proposition B.2 that \( A \) has two consecutive rows which together have at most \( 4c_k \) ones. We contract these two rows and repeat. In the following iterations, the error, or red entries are treated like 1s by the Marcus Tardos theorem. This is because such entries point to the existence of a 1-entry in the original matrix \( A \) which could be used to form a \( k \)-grid minor. Note that no row or column ever exceeds \( 4c_k \) red entries if the algorithm succeeds to give a contraction sequence. \( \square \)

Theorem 2.5. There is a computable function \( f \) with \( f(k) = 2^{O(k \log k)} \) such that the following holds. There is an algorithm that, given a 0-1 matrix \( A \), in \( f(k)n^{O(1)} \) time either
- finds a \( k \)-grid minor in \( A \), or
- certifies that \( gr(A) \leq f(k) \).

Proof. If \( A \) does not have a \( k \)-grid minor, then by Proposition B.3 there is a \( 4c_k \)-contraction sequence of \( A \) that respects the order of the rows and columns of \( A \). In the words of [3], this means that \( A \) is \( 4c_k \)-twin-ordered, and a theorem in [3] asserts that \( A \) cannot have a \( (8c_k + 2) \)-mixed minor, which in turn implies that \( gr(A) \leq (8c_k + 2) = 2^{O(k \log k)} \). \( \square \)