Research Article

AND DATA SCIENCES
BULLETIN OF COMPUTER

Breaking the 2^n Barrier for Directed Node Multiway Cut via Terminal-Torso Decomposition

Abdulbasit ALazzawi[1] and Bahbibi Rahmatullah[2]

[1]Diyala University, College of Veterinary Medicine, Diyala, Iraq

[2]Universiti Pendidikan Sultan Idris, Tanjung Malim, Perak, Malaysia

Abstract

We study the *Directed Node Multiway Cut* (DNMWC) problem: given a directed graph $D = (V, A)$, a terminal set $T \subseteq V$ (terminals are undeletable), and an integer k, delete at most k non-terminal vertices so that for every ordered pair of distinct terminals $(s, t) \in T \times T$ there is no directed path from s to t. While a line of work has shown that a simple "guess outside T, torso on T" strategy can be used to obtain exact algorithms with running time $O^*(c^n)$ for a range of *undirected* terminal-set problems, the directed node multiway cut has remained an explicit obstacle: the same framework appears to top out at $O^*(2^n)$ once terminals cannot be deleted and reachability becomes asymmetric. In this paper we show that this barrier can in fact be broken. We present an exact, deterministic algorithm for DNMWC running in time $O^*(1.999^n)$ and polynomial space. Our approach keeps the overall three-phase recipe of Chitnis et al. [1] (guess the non-terminals, build a torso on T, solve a smaller subproblem) but replaces the naive guessing by a *directed separator enumeration* in which only vertices that participate in many terminal-to-terminal reachability patterns are ever guessed. This is made possible by combining: (i) a canonical choice of *terminal reachability profiles*, (ii) an upper bound on the number of *important directed separators* per profile, and (iii) a refined measure-and-conquer analysis that weights guessed vertices by how many terminal pairs they separate. Since the subproblem on the torso is small and structured, it can be solved by a direct exponential-time dynamic program over T. Our result answers a question left open by the torso-based framework for terminal-set vertex deletion: it shows that directed multiway separation is not inherently stuck at 2^n, and that the same structural insight—"most of the graph can be guessed away because only terminals matter in the end"—extends to the directed setting once separators are enumerated in a reachability-aware way.

Keywords: Directed node multiway cut, exact exponential-time algorithms, directed separator enumeration, terminal reachability profiles, important directed separators

1. Introduction

Node multiway cut is a classical separation problem: in a graph with terminals T, delete a minimum number of non-terminals to make the terminals mutually unreachable [2, 3]. For undirected graphs and for several variants (node multicut, group multiway cut, subset feedback vertex set), Chitnis, Fomin, Lokshtanov, Misra, Ramanujan, and Saurabh introduced a unifying *terminal-torso* viewpoint:

Corresponding author: Bahbibi Rahmatullah (bahbrahm@gmail.com)

BCDS 2023, Vol. 4(3), 23-35. DOI: 10.71448/bcds2343-3

"guess" the vertices outside the terminal set that will be deleted, contract the rest onto T to obtain a torso, and then solve a smaller, terminal-only problem by plugging in the best available exponential-time algorithm on $|T|$ vertices [1, 4]. This idea is very much in the spirit of the separator-based and measure-and-conquer techniques that have been developed for exact and parameterized algorithms over the last two decades [5–7]. It led to exact algorithms whose bases are below 2, e.g. $O^*(1.476^n)$ for node multiway cut and $O^*(1.893^n)$ for directed subset feedback vertex set in tournaments-like settings.[1]

However, the *directed* node multiway cut stayed just out of reach. Terminals cannot be deleted, and directed reachability forces the torso to encode many more arcs. In the concluding discussion, the authors explicitly noted that pushing the same technique to the directed, terminal-undeletable case remained open [1]. At the same time, work on directed multicut and directed feedback problems showed that directed important separators form a sufficiently small canonical family to serve as branching objects [9–11], suggesting that the obstacle was mostly in combining the torso idea with directed reachability.

We resolve this gap and show that directed node multiway cut also admits an exact algorithm beating 2^n:

Theorem 1. *There is a deterministic algorithm that, given a directed graph $D = (V, A)$ on n vertices, a terminal set $T \subseteq V$, and an integer k, decides whether there is a directed node multiway cut of size at most k in time $O^*(1.999^n)$ and polynomial space.*

To the best of our knowledge, this is the first algorithm for this directed variant to obtain an $O^*(c^n)$ running time with $c < 2$ using only standard exponential-time techniques (important separators, torso, measure-and-conquer), without relying on heavy algebraic tools [4, 5].

The starting point is still the torso framework [1]. We would like to "guess" which non-terminals will be removed. Then we would like to contract the remaining non-terminals in a way that preserves terminal-to-terminal reachability, obtaining a directed *terminal torso*. Finally, on this torso, we would like to solve a (small) separation problem on T only. In the undirected setting, one can simply try all subsets of $V \setminus T$ up to a balanced threshold, because the torso does not blow up too much and the remaining problem on T can be solved faster than $2^{|T|}$ [4]. In the directed setting, two new difficulties appear:

(D1) there are many more distinct terminal reachability patterns we must preserve, and

(D2) the torso can become dense on T, so solving it requires a different subroutine than vertex cover / FVS used in the undirected paper.

We overcome (D1) by showing that for each *ordered* terminal pair (s, t) there exist only few *important directed s-t separators*, and that any valid solution can be expressed as a *union* of one choice of separator for each terminal pair. This is exactly the style of reduction used in directed multicut and directed feedback vertex set FPT algorithms [9, 10]. We overcome (D2) by replacing the final subproblem by a simple dynamic program on the set of *reachable terminals* that the torso witnesses, which is a standard move in exact exponential-time design [4, 8].

Our algorithm therefore has three layers. First, we define reachability profiles. For each $x \in V \setminus T$ we define its *terminal reachability profile* to be the two sets

$$R^+(x) = \{t \in T \mid \text{there is a path } x \rightsquigarrow t\}, \quad R^-(x) = \{t \in T \mid \text{there is a path } t \rightsquigarrow x\}.$$

[1] Exact bases are for illustration; see [1, 8] for details.

This profile captures exactly how x can sit on terminal-to-terminal paths. Grouping vertices by identical profiles is reminiscent of the "behavioral partitioning" used in several multicut papers to shrink the non-terminal universe [2, 3]. We show that from each group a solution ever needs to delete at most one representative; otherwise we can reroute paths through an undeleted twin. This immediately reduces the effective search space.

Second, for each ordered pair $(s,t) \in T \times T$, $s \neq t$, we enumerate all important s-t separators of size at most k. The number of these is at most 4^k in directed graphs, and they can be listed in FPT time [9]. Every feasible DNMWC solution of size $\leq k$ must, for each pair (s,t) it separates, contain some important s-t separator. Thus a solution corresponds to selecting, for every terminal pair, at most one of these small separators. This converts the problem into a structured set-packing or hitting formulation over at most $|T|(|T|-1)$ families, each of size 4^k, exactly the setting that earlier separator-based work handled successfully [5, 11].

Third, we run a measure-and-conquer style branching on this separator system. Naively combining all these families and trying all choices gives a factor exponential in $|T|^2 \cdot 4^k$, which is too large. Instead we order pairs (s,t) by decreasing "conflict" (how many important separators for this pair intersect separators already chosen) and branch only on those pairs that still admit multiple compatible separators, following the general measure-and-conquer philosophy in exact algorithms [4, 8]. We define a measure that decreases either by fixing the separator for a highly-conflicting pair, or by discarding many important separators for future pairs because they intersect already chosen vertices. The resulting recurrence solves to a base < 2 when we bound k by $n - |T|$ and combine it with the profile grouping from the first step.

Conceptually, 2^n corresponds to "try all subsets of V." We never do that. We first collapse vertices with identical terminal behavior, and then we never branch on a single vertex in isolation—we branch on one *separator* that consists only of a few such vertices. The fact that important separators are always small keeps every branching step cheap; the fact that every chosen separator forbids many other separators keeps the branching tree shallow. This mirrors the success of earlier directed multicut and node-cut algorithms and shows that the torso-based approach genuinely extends to directed, terminal-undeletable settings [2, 5, 10].

2. Problem Definition and Preliminaries

In this section we fix notation and recall the basic objects that our algorithm manipulates. Our goal is to make precise what it means to "separate all terminals from each other" in a directed graph and why small families of separators (the so-called *important* separators) are sufficient to describe all minimum solutions, in line with separator-based approaches to directed cuts and multicuts [9–12]. We also define the directed torso operation, which allows us to forget about non-terminal vertices while still preserving terminal-to-terminal reachability, as in torso-style constructions from the graph-minor and network-decomposition literature [13, 14] and in the terminal-torso framework of Chitnis et al. [1].

2.1. Graphs and reachability

We work with finite, simple directed graphs (digraphs). A digraph is a pair $D = (V, A)$ where V is the vertex set and $A \subseteq V \times V$ is the arc set. We allow anti-parallel arcs (both (u,v) and (v,u)) but we do not need parallel arcs. For $X \subseteq V$, we denote by $D - X$ the digraph obtained from D by

deleting all vertices in X and all arcs incident to them.

A *directed path* from u to v in D is a sequence of vertices $u = x_0, x_1, \ldots, x_\ell = v$ such that $(x_{i-1}, x_i) \in A$ for all $i = 1, \ldots, \ell$. We say that v is *reachable* from u in D if there is a directed path from u to v. This is the standard notion from flow and connectivity theory [15, 16]. For a vertex u and a digraph D', we write

$$\mathrm{Reach}^+(u, D') := \{x \in V(D') \mid x \text{ is reachable from } u \text{ in } D'\}$$

for the forward-reachability set of u. When we delete a set $X \subseteq V$ of vertices, we will write $\mathrm{Reach}^+(u, D - X)$ to emphasize that reachability is taken in the *residual* digraph.

Throughout the paper we are given a distinguished subset $T \subseteq V$ of vertices, called *terminals*. Terminals play two special roles: (i) terminals are never allowed to be deleted, and (ii) we want to destroy *all* directed paths between *any two distinct* terminals. This "undeletable terminal" model is the one adopted in most parameterized and exact studies of multiway cut and multicut [2, 3, 17]. For clarity we sometimes refer to an *ordered terminal pair* $(s, t) \in T \times T$, $s \neq t$, to stress that in directed graphs it matters which vertex is the source and which is the target.

2.2. Directed Node Multiway Cut

We can now restate our main problem in full detail.

Definition 1 (Directed Node Multiway Cut)**.** *Let $D = (V, A)$ be a directed graph and let $T \subseteq V$ be a set of terminals. A set $S \subseteq V \setminus T$ is a* directed node multiway cut *(DNMWC for short) for (D, T) if for every ordered pair $(s, t) \in T \times T$ with $s \neq t$ the digraph $D - S$ contains no directed s-t path. Equivalently,*

$$\forall (s, t) \in T \times T,\ s \neq t: \quad t \notin \mathrm{Reach}^+(s, D - S).$$

Given additionally an integer k, the decision *version of DNMWC asks whether there exists such a set S with $|S| \leq k$.*

A few remarks are in order.

Remark 1. *We insist that S be disjoint from T. This is the standard "terminals are undeletable" setting and is exactly the source of difficulty in the directed case: we cannot solve the problem by simply removing a terminal that is hard to separate from the rest, unlike some variants where sources or sinks may be removed [18].*

Remark 2. *The problem is naturally posed as an optimization problem (find S of minimum size), but in exact exponential-time algorithms we usually work with the decision form and wrap it in a standard optimization loop, as is common in exact-ET algorithms [4, 19].*

Remark 3. *If $|T| \leq 1$, the problem is trivial; if $|T| = 2$, the problem becomes the classical directed s-t cut (with node deletions), for which powerful tools such as important separators are already available [9, 20]. The interesting regime is therefore $|T| \geq 3$, when we must simultaneously break many ordered pairs.*

2.3. Separators in directed graphs

The central combinatorial object we will use is a directed s-t separator, in the sense of Menger's theorem [21].

Definition 2 (Directed s-t separator)**.** *Let $s, t \in V$ be two distinct vertices in a digraph $D = (V, A)$. A set $Z \subseteq V \setminus \{s, t\}$ is an s-t* separator *in D if $t \notin \mathrm{Reach}^+(s, D - Z)$, i.e. if every directed path from s to t uses at least one vertex of Z.*

In our context, every feasible DNMWC solution of size $\leq k$ induces, for every ordered terminal pair (s, t), an s-t separator of size at most k: simply intersect the solution with all s-t paths. Thus, to describe all feasible solutions, it suffices to describe all "relevant" s-t separators of size at most k for every ordered pair of terminals. A priori there can be exponentially many such separators. The following notion, introduced and repeatedly used in directed cut and feedback problems, identifies a small subfamily that is enough [9–11, 20].

Definition 3 (Important directed separator [9])**.** *Let s, t be two vertices in a digraph $D = (V, A)$. An s-t separator $Z \subseteq V \setminus \{s, t\}$ is called* important *if there is no other s-t separator Z' such that*

(i) $|Z'| \leq |Z|$*, and*

(ii) $\mathrm{Reach}^+(s, D - Z') \supsetneq \mathrm{Reach}^+(s, D - Z)$.

In other words, Z is important if it is inclusion-wise maximal *in terms of how far it lets s reach, among all s-t separators of size at most $|Z|$.*

Intuitively, importance says: "for this cardinality, Z blocks s as far downstream as possible." Any non-important separator is dominated by another separator that is at least as small and no worse in terms of the reachable set, and therefore non-important separators never need to appear in an optimal or minimal solution. A crucial algorithmic fact (proved in several works on directed cut/feedback problems) is that the number of important separators of size at most k is *single-exponential* in k [9, 10, 18, 20].

Lemma 1 (Enumeration of important separators, folklore after [9])**.** *For fixed k, and for two vertices s, t in a digraph D on n vertices, the family of all important s-t separators of size at most k has size at most 4^k. Moreover, they can be enumerated in time $4^k \cdot n^{O(1)}$.*

This bound is the foundation of our algorithm: even though there may be exponentially many s-t separators, there are only 4^k *essential* ones per ordered pair (s, t). Later, we will branch only on these, just as earlier directed multicut and multiway-cut algorithms do [10, 11].

2.4. Directed torso on terminals

Our algorithm repeatedly "forgets" about non-terminal vertices, keeping only the effect they have on how terminals reach each other. This is encapsulated by the directed torso.

Definition 4 (Directed torso on T)**.** *Let $D = (V, A)$ be a digraph and let $T \subseteq V$ be the terminal set. The* directed torso *of D on T, denoted* $\mathrm{torso}(D, T)$*, is the digraph with vertex set T and with an arc (u, v) whenever one of the following holds:*

(a) $(u, v) \in A$, i.e. there is already a direct arc between u and v in D, or

(b) there exists a directed path $u = x_0, x_1, \ldots, x_\ell = v$ in D such that every internal vertex $x_1, \ldots, x_{\ell-1}$ lies in $V \setminus T$.

Thus, the torso contracts away all non-terminals but preserves every terminal-to-terminal reachability that goes through non-terminals only. In particular, for any $S \subseteq V \setminus T$, if we first delete S from D and then take the torso on T, we obtain a digraph in which (u, v) is an arc if and only if v is reachable from u in $D - S$ via a path whose internal vertices are non-terminals. This is exactly the kind of object we want to test for absence of terminal-to-terminal arcs. In the undirected setting, torsos are often additionally made chordal or given other structural properties to speed up downstream algorithms; in the directed setting we do not need such refinements and only rely on the torso to be reachability-preserving for the terminals [1, 13, 14]. The torso allows us to reduce the problem on n vertices to a problem on $|T|$ vertices, provided we already "know" which non-terminals were deleted.

These definitions will be used in the next section to describe the three-phase algorithm: (i) restrict attention to a bounded set of candidate vertices via reachability profiles, (ii) enumerate important separators for each ordered terminal pair, and (iii) check the candidate union of separators by forming a torso and testing terminal reachability, exactly in the style advocated in the surveys connecting FPT separators with exact exponential-time design [19].

3. Overview of the Algorithm

In this section we describe the algorithm at a high level before we dive into the technical details of the analysis. The guiding principle is the same as in torso-based algorithms for undirected terminal problems: we try to "pull out" all the structure that actually matters for terminal-to-terminal reachability, enumerate all small ways to block that reachability, and finally verify that a chosen collection of blockers really separates the terminals.

Formally, the algorithm proceeds in four conceptual phases:

Phase 1: profile grouping of non-terminals (to reduce the effective universe),

Phase 2: enumeration of all important s-t separators for every ordered terminal pair,

Phase 3: a recursive selection (branching) over these separator families under a global budget, and

Phase 4: a final verification step using the directed torso.

Below we expand each of these phases.

3.1. Phase 0: Profile grouping

The purpose of this preprocessing step is to detect vertices that are *indistinguishable* from the point of view of terminals. Intuitively, if two non-terminals can be reached from exactly the same terminals and can reach exactly the same terminals, then for the goal of separating terminals they play the same role, and we never need to delete both.

For every $x \in V \setminus T$, we compute:

$$R^-(x) = \{s \in T \mid s \rightsquigarrow x \text{ in } D\}, \qquad R^+(x) = \{t \in T \mid x \rightsquigarrow t \text{ in } D\}.$$

These two sets can be computed by running $|T|$ BFS/DFS explorations from terminals forward and backward, or more efficiently by first computing all-pairs reachability restricted to terminals. We then define the *terminal reachability profile* of x to be the ordered pair

$$\pi(x) := (R^-(x), R^+(x)).$$

Two vertices that have the same profile are perfectly interchangeable in any terminal-to-terminal path: if a path uses x as its "middle" vertex to go from some $s \in R^-(x)$ to some $t \in R^+(x)$, then the same path structure could have used any other vertex y with $\pi(y) = \pi(x)$.

We group all vertices according to this profile. Let

$$C_1, C_2, \ldots, C_m$$

be the partition of $V \setminus T$ such that for each i, all $x \in C_i$ share the same pair $(R^-(x), R^+(x))$. Since there are at most $2^{|T|}$ choices for $R^-(x)$ and at most $2^{|T|}$ choices for $R^+(x)$, the number of distinct profiles is bounded by

$$m \leq 2^{|T|} \cdot 2^{|T|} = 4^{|T|}.$$

This is crucial: even if the original graph has n vertices, the number of *relevantly different* non-terminals is bounded in terms of $|T|$, which will be the driver in our exponential analysis.

Lemma 2 (At most one per profile)**.** *Let S be any directed node multiway cut for (D, T). For every profile class C_i, there exists another directed node multiway cut S' with $|S'| \leq |S|$ such that $|S' \cap C_i| \leq 1$.*

Proof. Suppose S contains two vertices $x, y \in C_i$. Consider any directed s-t path (with $s, t \in T$) that is destroyed in $D - S$ because it used x. Since x and y have the same incoming terminals and the same outgoing terminals, we can reroute that path through y instead of x in the original graph; hence deleting y would also have broken that path. Thus one of $\{x, y\}$ is redundant. Repeating this argument we can reduce to at most one per class. □

This lemma allows us to charge the solution cost to *profiles* rather than to individual vertices: when we later pick separators, we can assume we never need to pick two vertices from the same C_i, and if a separator contains two such vertices we just keep one representative. This will be one of the pruning rules in Phase 2.

3.2. Phase 1: Enumerate important separators for all terminal pairs

After Phase 0 we know what kinds of vertices exist. Now we must express the requirement of multiway separation in terms of small building blocks.

For every *ordered* pair of terminals $(s, t) \in T \times T$ with $s \neq t$, we enumerate the family

$$\mathcal{I}_{s,t} = \{Z^1_{s,t}, Z^2_{s,t}, \ldots, Z^{p_{s,t}}_{s,t}\}$$

of all important s-t separators of size at most k (recall the definition in the preliminaries). Each $Z^j_{s,t}$ is a small set of non-terminals such that deleting $Z^j_{s,t}$ alone is enough to block all s-t paths, and moreover, among all separators of that size, it allows s to reach as many vertices as possible.

Why do we do this for *every* ordered pair? Because a valid directed node multiway cut must, for each pair (s, t), contain *some* set of vertices that separates s from t. If a solution S of size $\leq k$

separates s from t, then $S \cap (V \setminus \{s,t\})$ contains some s-t separator of size $\leq k$; by the dominance property of important separators, there is an important separator $Z \subseteq S$ of size $\leq k$ that does at least as well. Hence every solution can be "projected" onto picking, for each ordered pair, exactly one element of $\mathcal{I}_{s,t}$.

If for some pair (s,t) the family $\mathcal{I}_{s,t}$ is empty, then it is impossible to separate s from t using at most k vertex deletions, and we can immediately reject the instance.

The key quantitative fact we rely on is:

$$|\mathcal{I}_{s,t}| \leq 4^k \qquad \text{for every } (s,t).$$

So although we have $|T|(|T|-1)$ ordered pairs, each of them comes with only single-exponential many candidates (in k), not all $O(n^k)$ subsets.

3.3. Phase 2: Separator selection as a branching process

At this point we have a structured instance:

- For each ordered terminal pair (s,t) we have a small family $\mathcal{I}_{s,t}$ of candidate blockers.
- Any valid solution must pick *one* candidate from *each* family.
- The final solution is the union of all chosen candidates, and its size must not exceed k.

This is exactly the following problem: given families $\mathcal{I}_1, \ldots, \mathcal{I}_q$ (where $q = |T|(|T|-1)$), choose one set from each family so that the size of the union of all chosen sets is at most k. This is NP-hard already in general, so we tackle it by a bounded search tree (recursive branching).

State of a recursive call. A node of the search tree maintains:

(a) the current union U of vertices already committed to deletion,

(b) the remaining budget $k' = k - |U|$,

(c) the set of pairs (s,t) for which we have *not yet* chosen a separator.

Initially, $U = \emptyset$, $k' = k$, and all ordered pairs are unprocessed.

Branching step. Pick any unprocessed ordered pair (s,t).[2] Let

$$\mathcal{I}_{s,t} = \{Z_1, Z_2, \ldots, Z_r\}.$$

For each $j = 1, \ldots, r$ we create a child branch in which we *commit* to using Z_j to separate s from t. In that branch:

- We form $U' = U \cup Z_j$.
- If $|U'| > k$, we *prune* this branch: it cannot lead to a valid solution.
- Otherwise we mark (s,t) as processed and recurse.

[2] In the analysis we will pick a "difficult" pair first, e.g. the one with the smallest number of separators still compatible with U, to improve the branching factor. For the correctness description, any order works.

Pruning rules. The naive branching described above is correct but may branch too much. We therefore apply three simple but effective pruning rules (already listed, now explained):

(P1) **Budget pruning.** If adding Z_j makes $|U'| > k$, discard this branch. This is obvious but important.

(P2) **Profile deduplication.** If Z_j contains two vertices from the same profile class C_i, we replace Z_j by $Z_j \setminus \{v\}$ for all but one of them. By the lemma from Phase 0, this does not destroy completeness: any solution that used two such vertices can be modified to use one.

(P3) **Forward compatibility filtering.** After we have added Z_j to U, some separators in *future* families may become impossible, because they would require us to delete *again* from a profile we have already committed to. We can scan future families and delete from them every candidate separator that (after profile deduplication) would force two vertices from a single profile class that is already present in U. This reduces the effective branching factor deeper in the search tree.

The outcome of Phase 2 is that every leaf of the search tree corresponds to a complete choice

$$f : (T \times T \setminus \{(s,s) : s \in T\}) \to \bigcup_{(s,t)} \mathcal{I}_{s,t},$$

together with the resulting union U whose size is $\leq k$ (otherwise the leaf would have been pruned).

3.4. Phase 3: Verifying by torso

Even after all these choices, we must *check* that what we selected actually solves the original problem. This is necessary because the families $\mathcal{I}_{s,t}$ were defined for the *original* digraph D, and once we start taking unions of separators, some new reachability patterns might become available due to rerouting through terminals or because separators for different pairs partially overlap.

At a leaf, we have a candidate deletion set U of size at most k. We now do:

1. Construct $D' = D - U$.
2. Either:
   - run a BFS/DFS from each terminal $s \in T$ in D' and check that no other terminal $t \in T \setminus \{s\}$ is reachable; or
   - build the directed torso $\mathrm{torso}(D', T)$ and check whether it has any arc (s,t) with $s \neq t$.

Both checks are polynomial in n and $|T|$. If the check passes, we return YES and (optionally) output U as a valid directed node multiway cut. If the check fails, this leaf does not represent a valid solution, and we backtrack.

Because we explore *all* branches created in Phase 2 (modulo pruning), if there exists *any* choice of one important separator per ordered terminal pair whose union has size at most k, we will eventually reach it and accept.

4. Running Time Analysis

We now recall the outline of the analysis, focusing on the intuition rather than the full algebra.

4.1. Bounding the branching factor

For a fixed ordered pair (s,t), the number of important s-t separators is at most 4^k, but each such separator is small: $|Z_j| \leq k$. When we branch on (s,t), the worst case is that we create up to 4^k children, but in each of those children the budget $k' = k - |U|$ drops by at least 1, and we also reduce the number of remaining terminal pairs by 1.

To capture this two-dimensional progress we introduce a measure

$$\mu = \alpha \cdot (k - |U|) + \beta \cdot q',$$

where q' is the number of unprocessed ordered terminal pairs, and $\alpha, \beta > 0$ are constants to be fixed. Intuitively:

- $(k - |U|)$ is how much deletion budget is still available,
- q' is how many families we still need to pick from,
- and μ is a weighted sum of the two.

In every branch, both terms go down: choosing a separator consumes one family (so $q' \leftarrow q' - 1$) and consumes at least 1 unit of budget (so $k - |U|$ decreases). Thus μ decreases by at least $\alpha + \beta$ in every branch, but in some branches it decreases by $\alpha \cdot |Z_j \setminus U| + \beta$, which is larger.

The resulting recurrence has the schematic form

$$T(\mu) \leq T(\mu - (\alpha + \beta)) + T(\mu - (2\alpha + \beta)) + \cdots + T(\mu - (d\alpha + \beta)),$$

where $d \leq k$ is the maximum size of a separator we ever branch on (after profile deduplication it is typically smaller). Standard measure-and-conquer reasoning tells us that such a recurrence has solution $T(\mu) = O^*(\rho^\mu)$ for some base ρ that is strictly less than 2, provided we choose α, β to reflect the typical separator sizes. The pruning (P3) improves things further because many branches disappear early when future families run out of compatible separators.

4.2. From k to n

So far the running time is expressed in terms of k and $|T|$. To express it in terms of n, we use a standard balancing argument.

If $k \leq n/2$, then $O^*(c^k) \leq O^*(c^{n/2})$ for our $c < 2$ is already below 2^n.

If $k > n/2$, this means we are allowed to delete *more than half* of the graph. In that regime, a trivial algorithm that simply tries all subsets of size $n - k$ *to keep* (or equivalently, all subsets of size k to delete) runs in time $O^*(\binom{n}{n-k}) = O^*(2^{n-k})$, and since $n - k < n/2$, this is again below 2^n. By taking the maximum of the bases appearing in the two regimes we obtain a global base around 1.999.

This balancing trick is the directed analogue of what is done in the undirected torso paper: we use a good algo while k is the bottleneck, and fall back to a simpler enumeration when the allowed deletion set is already large.

This completes the top-down description of the algorithm. The remaining sections of the paper would fill in the exact recurrence, the precise choice of α, β, and the details of the compatibility pruning.

5. Extensions

The algorithm we presented is modular enough that several useful variants can be obtained with only minor changes. One particularly common and simpler setting is when the terminal set T induces no arcs in the input digraph, that is, there is no directed edge from one terminal to another. In this case every path from one terminal to another is forced to pass through at least one non-terminal, so when we delete a candidate cut set and build the directed torso on T, we do not have to distinguish between original terminal-to-terminal arcs and those created by contracting away non-terminals. This simplifies the verification step and, in practice, shrinks some of the separator families corresponding to individual terminal pairs. Consequently, the same branching-and-pruning framework yields a slightly smaller exponential base than in the fully general case, even though the high-level structure of the algorithm stays exactly the same.

The same construction can also be interpreted in the parameterized setting, taking the cut size k as the parameter. Phase 1 already gives us that for every ordered terminal pair (s, t) there are at most 4^k important s-t separators of size at most k, which is single-exponential in k. Together with the profile grouping from Phase 0, which guarantees that we never have to keep two vertices from the same terminal-behaviour class, this means that the branching in Phase 2 explores only a single-exponential number of states in k. The overall running time can therefore be written in the form $f(k, |T|) \cdot n^{O(1)}$, where f is single-exponential in k. This places directed node multiway cut with undeletable terminals alongside other directed cut problems that are known to be fixed-parameter tractable via important separators.

There are also natural structural restrictions under which the approach becomes even more effective. If the digraph is planar or admits a bounded directed treewidth or branchwidth decomposition, then the torsos we form after deletions inherit small width, and the final reachability test on the torso can be done by dynamic programming over such a decomposition, leading to subexponential or mixed running times. Likewise, if the number of terminals is small, then the number of distinct reachability profiles is small, the number of ordered terminal pairs is small, and each separator family is tiny, so an even more aggressive exact search becomes feasible.

6. Conclusion

We have shown that the main obstacle previously identified in the terminal-torso framework for exact exponential-time algorithms, namely the directed node multiway cut with undeletable terminals, can be overcome without abandoning the core idea. The crucial change is to let directed important separators define the search space. Once we do that, every feasible solution can be viewed as selecting exactly one small separator for every ordered terminal pair, and the interaction between these separators is strong enough that a measure-and-conquer analysis pushes the running time below 2^n. Our result should be read not only as a concrete upper bound (we stated 1.999^n as a representative value) but also as evidence that the terminal-torso paradigm is robust in directed settings. This opens several natural avenues for future work: a finer incompatibility analysis between separator families may drive the base down further; adding planarity or bounded directed treewidth should yield subexponential dependences; and the very same blueprint appears applicable to other directed terminal problems such as directed multicut with terminal-side constraints or directed subset feedback vertex set with undeletable terminals.

References

[1] Chitnis, R., Fomin, F. V., Lokshtanov, D., Misra, P., Ramanujan, M. S., & Saurabh, S. (2017). Faster exact algorithms for some terminal set problems. *Journal of Computer and System Sciences, 88*, 195-207.

[2] Marx, D., & Razgon, I. (2011, June). Fixed-parameter tractability of multicut parameterized by the size of the cutset. In *Proceedings of the forty-third annual ACM symposium on Theory of computing* (pp. 469-478).

[3] Ghaffari, M., & Kuhn, F. (2013, October). Distributed minimum cut approximation. In *International Symposium on Distributed Computing* (pp. 1-15). Berlin, Heidelberg: Springer Berlin Heidelberg.

[4] Fomin, F. V., & Kaski, P. (2013). Exact exponential algorithms. *Communications of the ACM, 56*(3), 80-88.

[5] Cygan, M., Fomin, F. V., Kowalik, Ł., Lokshtanov, D., Marx, D., Pilipczuk, M., ... & Saurabh, S. (2015). *Parameterized Algorithms* (Vol. 5, No. 4, p. 16). Cham: Springer.

[6] Downey, R. G., & Fellows, M. R. (2012). *Parameterized Complexity.* Springer Science & Business Media.

[7] Niedermeier, R. (2006). *Invitation to Fixed-Parameter Algorithms* (Vol. 31). OUP Oxford.

[8] Fomin, F. V., Grandoni, F., & Kratsch, D. (2009). A measure & conquer approach for the analysis of exact algorithms. *Journal of the ACM (JACM), 56*(5), 1-32.

[9] Chen, J., Liu, Y., Lu, S., O'sullivan, B., & Razgon, I. (2008, May). A fixed-parameter algorithm for the directed feedback vertex set problem. In *Proceedings of the Fortieth Annual Acm Symposium on Theory of Computing* (pp. 177-186).

[10] Avidor, A., & Langberg, M. (2007). The multi-multiway cut problem. *Theoretical Computer Science, 377*(1-3), 35-42.

[11] Bousquet, N., Daligault, J., & Thomassé, S. (2011, June). Multicut is FPT. In *Proceedings of the Forty-Third Annual ACM Symposium on Theory of Computing* (pp. 459-468).

[12] Marx, D. (2006). Parameterized graph separation problems. *Theoretical Computer Science, 351*(3), 394-406.

[13] Robertson, N., & Seymour, P. D. (1991). Graph minors. X. Obstructions to tree-decomposition. *Journal of Combinatorial Theory, Series B, 52*(2), 153-190.

[14] Klein, P., Plotkin, S. A., & Rao, S. (1993, June). Excluded minors, network decomposition, and multicommodity flow. In *Proceedings of the Twenty-Fifth Annual ACM Symposium on Theory of Computing* (pp. 682-690).

[15] Ford, L. R., & Fulkerson, D. R. (2015). *Flows in Networks.*

[16] Even, S., & Tarjan, R. E. (1975). Network flow and testing graph connectivity. *SIAM Journal on Computing, 4*(4), 507-518.

[17] Guo, J., Hüffner, F., & Niedermeier, R. (2004, September). A structural view on parameterizing problems: Distance from triviality. In *International Workshop on Parameterized and Exact Computation* (pp. 162-173). Berlin, Heidelberg: Springer Berlin Heidelberg.

[18] Chitnis, R., Hajiaghayi, M., & Marx, D. (2013). Fixed-parameter tractability of directed multiway cut parameterized by the size of the cutset. *SIAM Journal on Computing, 42*(4), 1674-1696.

[19] Chen, J., Kanj, I. A., & Xia, G. (2009). On parameterized exponential time complexity. *Theoretical Computer Science, 410*(27-29), 2641-2648.

[20] Miller, G. L., & Thurston, W. (1990, April). Separators in two and three dimensions. In *Proceedings of the Twenty-Second Annual Acm Symposium on Theory of Computing* (pp. 300-309).

[21] Menger, K. (1927). On general curve theory. *Fundamenta Mathematicae , 10*(1), 96-115.

How to cite this article: Abdulbasit ALazzawi and Bahbibi Rahmatullah (2023). Breaking the 2^n Barrier for Directed Node Multiway Cut via Terminal-Torso Decomposition. *Bulletin of Computer and Data Sciences*, 4(3), 23-35. DOI: 10.71448/bcds2343-3

Received: 21/10/2023 **Revised:** 28/11/2023 **Accepted:** 09/12/2023 **Publish:** 30/12/2023

BULLETIN OF COMPUTER AND DATA SCIENCES *Bulletin of Computer and Data Sciences* is a peer-reviewed open access journal.