Solving Directed Multiway Cut Faster Than 2^n

Mingyu Xiao

School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu, China

Abstract

In the Directed Multiway Cut problem, we are given a directed graph $G = (V, E)$ and a subset $T \subseteq V$, called the terminal set. The aim is to find a minimum sized set $S \subseteq V \setminus T$, such that after deleting S, no directed path exists from one terminal to another terminal in the remaining graph. It has been an open question whether Directed Multiway Cut can be solved faster than the trivial running-time bound $O^*(2^{|V|})$. In this paper, we provide a positive answer to this question, presenting an algorithm with a running-time bound $O(1.9967^{|V|})$.

2012 ACM Subject Classification Theory of computation → Parameterized complexity and exact algorithms

Keywords and phrases Exact Algorithms, Parameterized Algorithms, Directed Multiway Cut, Directed Multicut, Directed Graphs

Digital Object Identifier 10.4230/LIPIcs.ESA.2024.104

Funding Mingyu Xiao acknowledges the National Science Foundation of China under Grant No. 62372095.

1 Introduction

The renowned min-cut max-flow theorem establishes that the size of the maximum flow equates to the size of the minimum cut between two vertices in a graph. Remarkably, both the maximum flow and the minimum cut can be computed in polynomial time, showcasing a compelling duality in combinatorics and algorithms. However, this elegant symmetry fails when we extend to a more generalized problem – one that involves separating more than two vertices from each other. According to Mader's Theorem [16], we encounter a somewhat weaker property in this context: given a subset T of vertices in an undirected graph G, there either exist k vertex-disjoint paths connecting two distinct vertices in T, or there is a vertex set of size at most $2k$ whose removal results in the disconnection of every pair of vertices within T. Even though a maximum set of vertex-disjoint paths connecting vertices in T can be computed efficiently in polynomial time using matching techniques, the corresponding cut problem, referred to as the Multiway Cut problem [5, 20], presents significant computational challenges. In the context of undirected graphs, Multiway Cut becomes NP-complete when the number of terminals in T is three [5]. For directed graphs, the problem turns even more intricate, becoming NP-complete with only two terminals [13]. This paper will delve into the exploration of exact exponential algorithms for the Directed Multiway Cut problem, which is formally defined as follows.

Directed Multiway Cut
Input: Directed graph $G = (V, E)$, a set $T = \{t_1, t_2, \ldots, t_l\}$ of l vertices in V, called terminals, and an integer k.
Output: Can we delete a set $S \subseteq V \setminus T$ of at most k vertices such that in the remaining graph $G - S$, there is no path between any two terminals in T?

In the case of undirected graphs, the problem is denoted as Undirected Multiway Cut. It is worth noting that Multiway Cut is alternatively referred to as Multiterminal Cut in various literature sources [5, 20]. Another extension of the minimum cut problem is known as Multicut [1, 15]. This problem, given a set of terminal pairs, is to disconnect all these terminal pairs by deleting the fewest number of vertices. Depending on whether the graph is directed or undirected, we encounter two distinct problems: Directed Multicut and Undirected Multicut. Multiway Cut is a specific instance of Multicut. In the former, each pair of terminals will constitute one terminal pair (or two terminal pairs for directed graphs) in the latter. Both Multiway Cut and Multicut represent significant generalizations of the minimum cut problem, drawing certain attention in the field of graph algorithms.

In the realm of exact algorithms, our goal is not to devise polynomial algorithms for NP-complete problems. Instead, we are interested in determining the fastest possible algorithm within the context of exponential algorithms. For a myriad of NP-complete problems, there are straightforward $O^*(2^n)$-time brute-force algorithms, which enumerate all vertex subsets for graph problems or all assignments for SAT problems, where n is the number of vertices in the graph or the number of variables in SAT. However, despite decades of research, no breakthrough has been made in surpassing this basic running-time bound for several fundamental NP-complete problems. The first NP-complete problem, SAT, is even subject to the SETH, which stipulates that no algorithm can solve it in $O(1.9999^n)$-time [14]. The well-known TSP has not seen any improvement beyond the $O^*(2^n)$-time dynamic programming method developed in the 1960s. Multiway Cut has also been extensively investigated in exact and parameterized algorithms. For Undirected Multiway Cut, Cygan et al. [4] introduced an $O^*(2^k)$-time algorithm, an improvement over the previous result of $O^*(4^k)$ [2]. Using the general framework of designing exact algorithms via parameterized algorithms in [8], we can directly solve Undirected Multiway Cut in $O^*(1.5^n)$ time. Subsequently, Chitnis et al. [3] proposed an $O(1.4767^n)$-time[1] algorithm for Undirected Multiway Cut. In the case of Undirected Multicut, Lokshtanov, Saurabh, and Suchý [15] broke the barrier 2^n for the first time by providing an $O(1.987^n)$-time algorithm. When the graph is a directed one, both Directed Multiway Cut and Directed Multicut become considerably more challenging. Whether they can be solved faster than $O(2^n)$ has remained an open question in the field of exact algorithms [3, 15]. In this paper, we focus on Directed Multiway Cut and present a non-trivial algorithm with a running-time bound of $O(1.9967^n)$.

Directed Multiway Cut admits a 2-factor approximation in polynomial time [17]. By using the Approximate Monotone Local Search framework described by Esmer et al. [7], we can get an algorithm for Directed Multiway Cut with the running time better than 2^n and the approximation ratio arbitrarily close to 1 (but not 1). The approximation factor can be made arbitrary small at the expense of increasing the running time but still keeping it the form of $(2-\epsilon)^n$ for some $\epsilon > 0$ whenever the approximation factor is strictly greater than 1. However, our result is still the first proof that the barrier of 2 for Directed Multiway Cut can be exactly broken.

We should also note the existence of unrestricted versions of Multiway Cut and Multicut where terminals are permitted to be included in the solution set S. For these unrestricted versions, the size k of the solution set cannot exceed the number l of terminals; otherwise, we would be better off simply deleting all terminals. This distinction is crucial. When the number l of terminals is small, a simple algorithm that enumerates all vertex subsets of size at most l may already be efficient. Conversely, when the number l of terminals

[1] We use the O-notation to hide the constant factor and the O^*-notation to hide the polynomial factor. Here we use $O(1.4767^n)$ instead of $O^*(1.4767^n)$ because the actual bound is $O^*(a^n)$ with $a = 1.4766\cdots < 1.4767$. Thus, $O^*(a^n) \subseteq O(1.4767^n)$.

is large, we can first enumerate the vertex subsets of $V \setminus T$ and simplify the problem. By striking a balance between $|T|$ and $|V| - |T|$, Chitnis et al. [3] demonstrated that the unrestricted version of Undirected Multiway Cut can be solved in $O(1.4767^n)$ time, an improvement over the previous result of $O(1.8638^n)$ in [11]. Furthermore, the unrestricted version of Directed Multiway Cut can be solved in $O(1.6181^n)$ time [3].

2 Preliminaries

Let $G = (V, E)$ denote a directed graph with a set V of vertices and a set E of edges. An edge in E is an ordered pair of vertices in V. Let $n = |V|$. We allow two edges with different directions between two vertices in the graph.

For a vertex $v \in V$, the set of *out-neighbors* of v is defined as $N^+(v) := \{u \in V : vu \in E\}$, and the set of *in-neighbors* of v is defined as $N^-(v) := \{u \in V : uv \in E\}$. For a vertex subset $V' \subseteq V$, let $N^+(V') = \bigcup_{v \in V'} N^+(v) \setminus V'$, $N^-(V') = \bigcup_{v \in V'} N^-(v) \setminus V'$, and $N(V') = N^+(V') \bigcup N^-(V')$. For a vertex subset U, we also let $N^+_U(V') = N^+(V') \bigcap U$, $N^-_U(V') = N^-(V') \bigcap U$, and $N_U(V') = N(V') \bigcap U$. The subgraph of G induced by a vertex subset $V' \subseteq V$ is denoted by $G[V']$, and we also use $G - V'$ to denote the subgraph $G[V \setminus V']$. A *path* of length ℓ is a sequence of vertices $(x_1, x_2, \ldots, x_{\ell+1})$ such that $x_i x_{i+1} \in E$ for each $1 \le i \le \ell$. If there exists a path from vertex u to vertex v, then we say vertex u can *reach* vertex v, and conversely, vertex v is *reachable from* vertex u.

In our problem, the set of terminals will be denoted by $T = \{t_1, t_2, \ldots, t_l\}$. A path is called a *$T$-path* if the first and last vertex are two distinct terminals. A vertex set is called a *multiway cut* if there is no T-path in the remaining graph after deleting them. In our problem, the target is to check whether there is a multiway cut of size at most k. We will be utilizing the following two simple properties in our algorithm.

▶ **Observation 1.** *Let v be a vertex not in any T-path in the graph G. A vertex set S is a multiway cut in G if and only if S is a multiway cut in the resulting graph after deleting v from G.*

▶ **Observation 2.** *Let G be a directed graph and G' be the graph after reversing the direction of all edges in G. A vertex set S is a multiway cut in G if and only if S is a multiway cut in G'.*

We will utilize Observation 1 to simplify the graph by removing certain vertices not involved in any T-path. Note that our algorithm may not delete all such vertices. We intentionally leave some in place to preserve certain properties of the graph structure, which will be explained later. For the sake of presentation, in one step of our algorithm, we will reverse the direction of all edges, and Observation 2 guarantees the correctness of this step.

3 The Algorithm for Directed Multiway Cut

The major contribution of this paper is to break the barrier 2^n for Directed Multiway Cut. We present our algorithm in this section. Most of the steps in our algorithm are simple branching rules. To verify the correctness of a branching rule, we only need to demonstrate that solutions exist if and only if at least one can be found in some subbranch. We will first introduce the main concept and the framework of the algorithm. Then, we will detail the specific steps involved in the algorithm. While introducing each branching rule, we also discuss its correctness and analyze its complexity.

3.1 The Framework

We now present the main concept of our algorithm. In our procedure, we consistently establish a vertex subset A that fulfills the following property.

Basic Property.

(1) There is exactly one terminal in A, which is always denoted by t;
(2) There is a path from terminal t to each other vertex in A;
(3) Vertices in A are not allowed to be included in the solution set S.

The set A determines a partition (A, B, R) of the vertex set of the graph. The set B is always the set of out-neighbors of A, i.e., $B = N^+(A)$. All the remaining vertices are in R, i.e., $R = V \setminus (A \cup B)$. See Fig. 1(a) for an illustration of the partition. The algorithm will always maintain this partition. Initially, the set A contains an arbitrary terminal $t \in T$. In most steps, the algorithm will select one vertex $v \in B$ and branch on v by either excluding it from the solution set S or including it in the solution set S. In the former branch, we include v in A, and then some vertices in R will move to B. In the latter branch, we delete v from the graph and decrease k by 1. The vertices in R may be iteratively moved to B or disconnected from $A \cup B$. By the Basic Property of A, we know that there is no terminal other than t in $A \cup B$; otherwise, the instance is a no-instance, and thus we can directly solve the problem when $R = \emptyset$. However, when $N^+_R(B) = \emptyset$, we may not be able to move vertices in R to B anymore. For this case, we will consider two subcases. If it also holds that $N^-(A \cup B) = \emptyset$, we delete $A \cup B$ from the graph since no vertex in $A \cup B$ is in a T-path, and select an arbitrary terminal in the remaining graph as a new initial terminal in A. If $N^-(A \cup B)$ is not an empty set, we will reverse the direction of all edges in the graph, keep set A the same and update $B = N^+(A)$. For this case, we can show that after deleting some vertices not in any T-path, the set A will also satisfy the Basic Property. Thus, we can continue our algorithm.

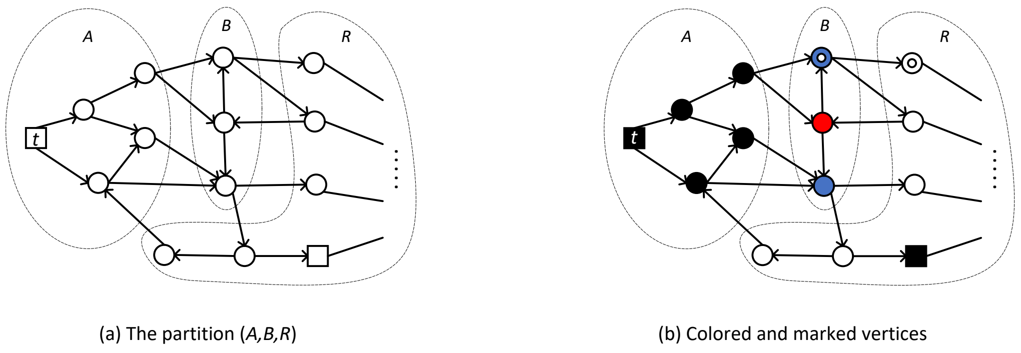

Figure 1 An illustration of the partition of the graph and colored and marked vertices, where a terminal is denoted by a square, a non-terminal vertex is denoted by a circle, and a small circle is put in each marked vertex.

In most steps, we just branch on a vertex by including it in the solution set or excluding it from the solution set, which may only lead to the trivial bound 2^n. Note that our algorithm will terminate when $A \cup B = V$ is the whole vertex set. At that time, the vertices in B are not branched on yet, which may save some running time. Keeping this in mind, we will use a measure-and-conquer method to analyze our algorithm and show that the running-time bound is strictly better than 2^n. The measure-and-conquer method was introduced by Fomin et al. [10]. It is a powerful tool to analyze exact algorithms. In the measure-and-conquer method, we will set weights to different types of vertices to distinguish their contribution to the complexity. To elucidate how we assign weights to the vertices, we use two different labels. The vertices will be colored in either black, red, or blue, meaning a vertex can be either one of these colors or uncolored. At a certain step in our algorithm, we will mark some vertices, making them either marked or unmarked. See Fig. 1(b) for a visualization of the vertices that have been colored and marked. We have five different weights for vertices: $1, \alpha = 0.958234, \beta = 0.943555, \gamma = 0.901789$, and 0. The weights of vertices with different labels are shown in Table 1.

Table 1 The setting of vertex weights, where $\alpha = 0.958234, \beta = 0.943555$, and $\gamma = 0.901789$.

	black	red	blue	uncolored
marked	0	γ	γ	β
unmarked	0	β	α	1

Let us first discuss the color label. Regardless of their marked status, black vertices are not permitted to be part of the solution set S, and their weight is therefore set to 0. According to the Basic Property, all vertices in A will be black. As per the problem definition, all terminals in T are also black. We may also generate additional black vertices in R. Vertices in B are colored either red or blue, based on the following definition of a *generalized out-neighbor*. The vertices in R can only be either black or uncolored. One step of our algorithm will also mark certain vertices. These marked vertices are reachable solely from one terminal t, i.e., there is no path from any terminal different from t to these vertices. However, not all vertices with this property will be marked.

Let w be the sum of the weight of all vertices in the graph. Then $w \leq n$. We will use w as the measure to analyze our algorithm and $C(w)$ to denote the worst size of the searching space of our algorithm on an instance with measure w. We will build a recurrence relation on $C(\cdot)$ for each branching operation. The complexity of the recurrence relation is determined by the corresponding branching factor. If among all the branching factors in the algorithm, the biggest one is μ, then the size of the searching tree is bounded by $O(\mu^w)$. For more detailed information on evaluating the size of the search tree and the time complexity bounds for recursive algorithms, refer to [12].

Generalized out-neighbor. A vertex u is referred to as a *generalized out-neighbor* of vertex v if vertex u is not colored black and there exists a path from v to u such that all vertices, except for v and u, within that path are colored black. The set of generalized out-neighbors of vertex v is denoted by $GN^+(v)$. For a vertex set V', we will also let $GN^+(V') = \bigcup_{v \in V'} GN^+(v)$. In our algorithm, we mainly consider generalized out-neighbors in set R. Thus, the following two sets will be frequently used: $GN^+_R(v) = GN^+(v) \bigcap R$ and $GN^+_R(V') = GN^+(V') \bigcap R$. A vertex $v \in B$ will be colored red if $GN^+_R(v) = \emptyset$ and blue if $GN^+_R(v) \neq \emptyset$. See Fig. 2 for an illustration of the generalized out-neighbors.

The concept of generalized out-neighbor is inspired by the idea of generalized neighbor for Feedback Vertex Set in undirected graphs used in [19, 9, 21].

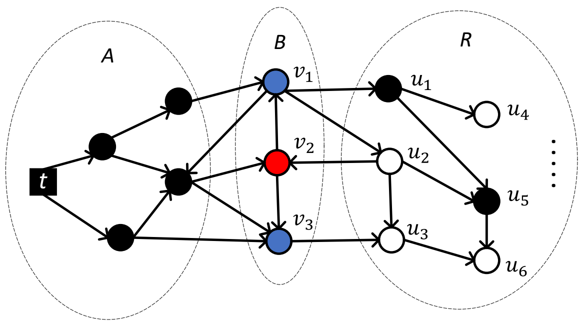

Figure 2 An illustration of generalized out-neighbors: in this graph, $GN_R^+(v_1) = \{u_2, u_4, u_6\}$, $GN_R^+(v_2) = \emptyset$, and $GN_R^+(v_3) = \{u_3\}$.

3.2 Main Steps of the Algorithm

In Directed Multiway Cut, terminals cannot be selected to the solution set. We assume that all terminals are colored black. We also assume that there is no edge between two terminals; otherwise, the instance is a no-instance. In the algorithm, when we say a vertex is not allowed to be selected to the solution set, we automatically color it black. Unless otherwise specified, after each operation of the algorithm, we also automatically update the color of vertices in $A \cup B$ by using the **Color Principle**: any vertex added to A is colored black; color an uncolored vertex $v \in B$ red if $GN_R^+(v) = \emptyset$ and blue if $GN_R^+(v) \neq \emptyset$. After most steps of the algorithm, the Basic Property of A trivially holds since we only extend A by moving one vertex from B to A in these steps. The Basic Property for only one step is not obvious, and we will give a proof. Thus, we will assume that the Basic Property always holds. We have four simple rules that will be iteratively applied to simplify the instance before executing any of the main steps.

▶ **Reduction Rule 1.** *If A is empty, include an arbitrary terminal $t \in T$ in A.*

▶ **Reduction Rule 2.** *If there is a black vertex in B, then move it to A immediately.*

This step will always keep the vertices in B red and blue.

▶ **Reduction Rule 3.** *If there is a vertex that can not reach any terminal, then delete it from the graph. If there is a vertex that is not reachable from any terminal, then delete it from the graph. If $N^+(A \cup B) = N^-(A \cup B) = \emptyset$, then delete $A \cup B$ from the graph.*

Whether a vertex can reach a terminal or is reachable from a terminal can be checked in polynomial time. A vertex that can not reach any terminal or is not reachable from any terminal is not in any T-path. Then we can delete it according to Observation 1. When $N^+(A \cup B) = N^-(A \cup B) = \emptyset$, no T-path includes a vertex in $A \cup B$. This is due to the fact that there is only one terminal t in A. Consequently, we can eliminate $A \cup B$ according to Observation 1 again.

Next, we are ready to introduce the main process of the algorithm. The algorithm is a recursive algorithm that contains nine branching steps described below. Before executing each branching step, we assume that we apply the above three reduction rules first and we do not describe this each time. Initially, the set A contains only a terminal t by Reduction Rule 1, B is $N^+(t)$, and $R = V \setminus (A \cup B)$. The algorithm will recursively do the following until $k < 0$ or the graph becomes an empty graph: If there is no marked vertex in $B \cup R$, execute Steps 1 to 4; if there is some marked vertex in $B \cup R$, execute Steps 5 to 9. Next, we introduce these steps.

▶ **Step 1.** *If there is a vertex $v \in B$ such that $|GN_R^+(v)| \geq 3$, then branch by either deleting v from the graph and decreasing k by 1 or including v in A.*

The correctness of Step 1 is trivial since any vertex v can be either in the solution or not. In Fig. 2, the vertex v_1 is a vertex satisfying the condition in Step 1. Let $d = |GN_R^+(v)|$. In the first branch, a blue vertex v is deleted from the graph, and the measure w decreases by α at least. In the second branch, a blue vertex v is moved to A, and it becomes black. Note that black vertices in $B = N^+(A)$ will be recursively moved to A by Reduction Rule 2. Thus, all vertices in $GN_R^+(v)$ will come to set B. This means that d uncolored vertices will be colored either red or blue. Note that $\alpha > \beta$. In total, the measure w decreases by $\alpha + d(1-\alpha)$ at least, where $d \geq 3$. We get a recurrence relation

$$C(w) \leq C(w-\alpha) + C(w-(3-2\alpha)), \tag{1}$$

the branching factor of which is 1.9736.

▶ **Step 2.** *If there is a vertex $v \in B$ such that $|GN_R^+(v)| = 1$, then assume $GN_R^+(v) = \{u\}$, and branch on u by either coloring u black or deleting u from the graph and decreasing k by 1.*

The correctness of Step 2 is also trivial. In Fig. 2, the vertex v_3 is a vertex satisfying the condition in Step 2, and after deleting vertex u_3 from the graph, the color of vertex v_3 will change from blue to red. In the first branch of this step, an uncolored vertex u is colored black, and the measure w decreases by 1. In the second branch, an uncolored vertex u is deleted from the graph, and the vertex v will become a vertex with $GN_R^+(v) = \emptyset$. Thus, its color will change from blue to red. The measure w decreases by at least $1 + (\alpha - \beta)$. We get a recurrence relation

$$C(w) \leq C(w-1) + C(w-(1+\alpha-\beta)), \tag{2}$$

the branching factor of which is 1.9900.

▶ **Step 3.** *If there is a vertex $v \in B$ such that $|GN_R^+(v)| = 2$, then assume $GN_R^+(v) = \{u_1, u_2\}$, and branch into three branches: coloring u_1 black; deleting u_1 from the graph, decreasing k by 1, and coloring u_2 black; deleting both of u_1 and u_2 from the graph and decreasing k by 2.*

The correctness of Step 3 is based on the following observation. Vertex u_1 can be in the solution set or not. In the branch where u_1 is in the solution, we can further branch on u_2 by either including it in the solution set or not. Thus, we get three branches. In the first branch of excluding u_1 from the solution set, an uncolored vertex u_1 is colored black, and the measure w decreases by at least 1. In the second branch, one uncolored vertex u_1 is deleted, one uncolored vertex u_2 is colored black, and the measure w decreases by at least 2. In the third branch, two uncolored vertices u_1 and u_2 are deleted, the color of vertex v changes from blue to red, and the measure decreases by at least $2 + (\alpha - \beta)$. We get a recurrence relation

$$C(w) \leq C(w-1) + C(w-2) + C(w-(2+\alpha-\beta)), \tag{3}$$

the branching factor of which is not greater than 1.9967.

When none of the above steps can be applied, it holds that $GN^+_R(B) = \emptyset$. After applying Reduction Rule 3, we also have that $N^+(A \cup B) = \emptyset$. If it further holds that $N^-(A \cup B) = \emptyset$, then there is no edge of any direction between $A \cup B$ and R. For this case, our algorithm will delete all vertices in $A \cup B$ by applying Reduction Rule 3. Furthermore, Reduction Rule 1 will be applied to add a new terminal t to A. Next, we assume that $N^-(A \cup B) \neq \emptyset$. We have some steps to handle with this case. Specially, we will mark some vertices and reverse the direction of edges in Step 4.

▶ **Step 4.** *If $GN^+_R(B) = \emptyset$ and $N^-(A \cup B) \neq \emptyset$, then we set all vertices in B as marked uncolored vertices, reverse the direction of all edges in the graph, keep the set A the same, and update the set B by letting $B = N^+(A)$ in the new graph.*

Note that after updating B in Step 4, we will automatically color the vertices in the new set B according to the **Color Principle**. We let Y denote the set of marked vertices not in A. After reversing the direction of all edges, the set $B = N^+(A)$ may change (even if we keep A the same). Thus, it may hold $B \neq Y$ after Step 4. An illustration of the graphs before and after Step 4 is shown in Fig. 3.

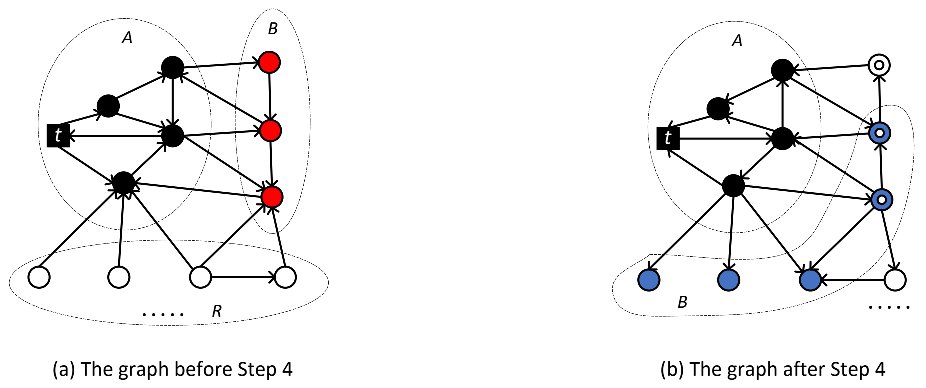

Figure 3 An illustration of the graphs before and after Step 4, where a small circle is put in each marked vertex.

We next show with a series lemmas culminating in Lemma 5 that the Basic Property of A still holds after Step 4.

▶ **Lemma 3.** *After executing Step 4, no terminal other than t can reach a vertex in $A \cup Y$.*

Proof. Let G^* and G denote the graph before and after executing Step 4 (reversing the direction of all edges), respectively. All vertices in B in G^* will be marked after Step 4. As we mentioned above that we execute Steps 1 to 4 only when there is no marked vertex in $B \cup R$. Thus, we know that in G the set Y of marked vertices not in A is exactly the set B in G^*. We have the condition $GN^+_R(B) = \emptyset$ in G^* when executing Step 4. Thus, in G^*, there is no directed path from a vertex in $A \cup B$ to any terminal in $T \setminus \{t\}$ since $A \cup B$ only contains one terminal t. Therefore, after reversing the direction of all edges, in G, no terminal in $T \setminus \{t\}$ can reach a vertex in $A \cup Y$. ◀

▶ **Lemma 4.** *After executing Step 4, for each vertex $u \in A \cup Y$, in the induced subgraph $G[A \cup Y]$, there is a path from terminal t to u and also a path from u to t. Furthermore, any path from terminal t to a vertex in $A \cup Y$ only contains vertices in $A \cup Y$.*

Proof. We use G^* and G to denote the graph before and after Step 4, respectively. First, we consider the graph G^*. The Basic Property of A holds in G^*, and hence there is a directed path from terminal t to each vertex $u \in A \cup B$ in the subgraph $G[A \cup B]$. Assume to the contrary that there is a vertex $u \in A \cup B$ such that there is no path from it to terminal t. Then u is not in any T-path to t. Furthermore, u is not in any T-path from t to a terminal in $T \setminus \{t\}$ since $GN_R^+(B) = \emptyset$. Therefore, vertex u would be deleted by Reduction Rule 3, a contradiction. We know that there is a path from each vertex $u \in A \cup B$ to terminal t in G^*. Furthermore, it is easy to derive a contradiction if in G there is a path from $u \in A \cup B$ to terminal t containing a vertex not in $A \cup B$. Thus, in G^*, for each vertex $u \in A \cup B$, there is a path from terminal t to u and also a path from u to t in the subgraph $G[A \cup B]$. Note that in G, the set A is the same and the set Y is the set B in G^*. The lemma holds. ◀

▶ **Lemma 5.** *After executing Step 4, the Basic Property of A still holds, and the measure w does not increase.*

Proof. Step 4 does not change the vertices in A and does not change the vertex color in A. After reversing the direction of all edges, by Lemma 4, we know that there is still a path from terminal t to each vertex $u \in A$. Thus, the Basic Property of A still holds.

Next, we consider the measure w. Step 4 first makes all vertices in Y as marked uncolored vertices. Before this, all vertices in Y are unmarked red or blue vertices. Thus, the weight of a vertex in Y will change from α or β to β according to Table 1. The measure will not increase. After updating set B, if a marked vertex is colored red or blue, its weight will change from β to γ; if an unmarked vertex is colored red or blue, its weight will change from 1 to α or β. In any case, the measure will not increase according to the value setting of α, β and γ in Table 1. We can conclude that the measure will not increase in Step 4. ◀

▶ **Lemma 6.** *After executing Step 4, for each marked vertex $v \in B$, there is a minimum multiway cut S^* such that either v is not in S^* or at least two vertices in $GN_R^+(v)$ are not in S^*.*

Proof. Let G be the graph. Assume that there is a minimum multiway cut S' in G such that $|GN_R^+(v) \setminus S'| \leq 1$ and $v \in S'$. We show that $S^* = (S' \cup GN_R^+(v)) \setminus \{v\}$ is another minimum multiway cut not containing v.

For each marked vertex $v \in B$, there is no path from a terminal in $T \setminus \{t\}$ to it by Lemma 3. Thus, each T-path containing v must be a path from t to another terminal t'. Let G' be the graph after deleting S^* from the graph. Assume to the contrary that there is a T-path P in G'. The path P must be a path from t to another terminal t'. Let P_1 be the subpath of P from v to the terminal t'. Since $GN_R^+(v) \subseteq S^*$ has been deleted in G', we know that P_1 must contain some other vertex in B. Let $u \in B$ be the last vertex in B on the path P_1. Let P_2 be the subpath of P_1 from u to the terminal t'. Since $u \in B$, we know that there is a path P_3 from t to u that contains only vertices in $A \cup \{u\}$. Then P_3 and P_2 will form a T-path from t to t'. This T-path does not contain v, and will also appear in the graph after deleting S' from G, which contradicts the fact that S' is a multiway cut in G. Thus, there is no T-path in G', and S^* is also a multiway cut.

Note that $|S^*| = |(S' \cup GN_R^+(v)) \setminus \{v\}| \leq |S'|$ since $|GN_R^+(v) \setminus S'| \leq 1$. Thus, S^* is a minimum multiway cut that does not contain v. ◀

By Lemma 6, we know that if $|GN_R^+(v)| \leq 1$, then there is always a minimum multiway cut not containing v.

▶ **Step 5.** *If there is a marked vertex $v \in B$ with $|GN_R^+(v)| \leq 1$, then move v to set A.*

In Step 5, a blue vertex v will be moved to A and colored black. We decrease the measure directly without branching. Based on Lemma 6, we can get the following branching rule for a marked vertex $v \in B$ with $|GN_R^+(v)| = 2$.

▶ **Step 6.** *If there is a marked vertex $v \in B$ with $|GN_R^+(v)| = 2$, then assume $GN_R^+(v) = \{u_1, u_2\}$, and branch by either moving v to set A or deleting v from the graph, decreasing k by 1, and coloring u_1 and u_2 black (including v in the solution set S but not u_1 and u_2).*

In the first branch, a marked blue vertex v will become a black vertex, and the measure w decreases by γ. In the second branch, one marked blue vertex v is deleted, and two uncolored vertices u_1 and u_2 are colored black. Note that the two vertices u_1 and u_2 may already be marked. Thus, the measure w decreases by at least $\gamma + 2\beta$. We get a recurrence relation

$$C(w) \leq C(w - \gamma) + C(w - (\gamma + 2\beta)), \tag{4}$$

the branching factor of which is 1.5166.

Next, we consider a marked vertex $v \in B$ with $|GN_R^+(v)| = 3$. By Lemma 6, we know that if a minimum multiway cut S^* contains v, then we can assume that S^* contains at most one vertex in $GN_R^+(v) = \{u_1, u_2, u_3\}$. For this case, the set S^* either does not contain u_1 or contains u_1 but does not contain u_2 and u_3.

▶ **Step 7.** *If there is a marked vertex $v \in B$ with $|GN_R^+(v)| = 3$, then assume $GN_R^+(v) = \{u_1, u_2, u_3\}$, and branch into three subbranches: moving v to set A; deleting v from the graph, decreasing k by 1, and coloring u_1 black (including v in the solution set S but not u_1); deleting v and u_1 from the graph, decreasing k by 2, and coloring u_2 and u_3 black (including v and u_1 in the solution set S but not u_2 and u_3).*

In the first branch, a marked blue vertex v becomes a black vertex, and the measure w decreases by γ. In the second branch, one marked blue vertex v is deleted and one (marked) uncolored vertex u_1 is colored black, and the measure w decreases by at least $\gamma + \beta$. In the third branch, one marked blue vertex v and one (marked) uncolored vertex u_1 are deleted and two (marked) uncolored vertices u_2 and u_3 are colored black, and the measure w decreases by at least $\gamma + 3\beta$. We get a recurrence relation

$$C(w) \leq C(w - \gamma) + C(w - (\gamma + \beta)) + C(w - (\gamma + 3\beta)), \tag{5}$$

the branching factor of which is 1.8453.

For a marked vertex $v \in B$ with $|GN_R^+(v)| = 4$ (assume $GN_R^+(v) = \{u_1, u_2, u_3, u_4\}$), similar to the above case, by Lemma 6, we can branch into four branches: $v \notin S$; $\{v, u_1\} \cap S = \{v\}$; $\{v, u_1, u_2\} \cap S = \{v, u_1\}$; $\{v, u_1, u_2, u_3, u_4\} \cap S = \{v, u_1, u_2\}$.

▶ **Step 8.** *If there is a marked vertex $v \in B$ with $|GN_R^+(v)| = 4$, then assume $GN_R^+(v) = \{u_1, u_2, u_3, u_4\}$, and branch into four subbranches: moving v to set A; deleting v from the graph, decreasing k by 1, and coloring u_1 black; deleting v and u_1 from the graph, decreasing k by 2, and coloring u_2 black; deleting v, u_1 and u_2 from the graph, decreasing k by 3, and coloring u_3 and u_4 black.*

In the first branch, a marked blue vertex v becomes a black vertex, and the measure w decreases by γ. In the second branch, one marked blue vertex v is deleted and one (possibly marked) uncolored vertex u_1 is colored black, and the measure w decreases by at least $\gamma + \beta$. In the third branch, one marked blue vertex v and one (possibly marked) uncolored vertex u_1 are deleted and one (possibly marked) uncolored vertex u_2 is colored black, and the measure w decreases by at least $\gamma + 2\beta$. In the forth branch, one marked blue vertex v and two (possibly marked) uncolored vertices u_1 and u_2 are deleted and two (possibly marked) uncolored vertices u_3 and u_4 are colored black, and the measure w decreases by at least $\gamma + 4\beta$. We get a recurrence relation

$$C(w) \leq C(w - \gamma) + C(w - (\gamma + \beta)) + C(w - (\gamma + 2\beta)) + C(w - (\gamma + 4\beta)), \tag{6}$$

the branching factor of which is not greater than 1.9967.

For a marked vertex $v \in B$ with $|GN_R^+(v)| \geq 5$, we just use a simple branching rule by either including it in the solution set or not.

▶ **Step 9.** *If there is a marked vertex $v \in B$ with $|GN_R^+(v)| \geq 5$, then branch by either moving v to set A or deleting v from the graph and decreasing k by 1.*

In the first branch, one marked blue vertex v becomes black, and at least $|GN_R^+(v)| \geq 5$ uncolored vertices will become red or blue. If an unmarked uncolored vertex becomes an unmarked red or blue vertex, the measure w decreases by at least $\min\{1 - \alpha, 1 - \beta\} = 1 - \alpha$. If a marked uncolored vertex becomes a marked red or blue vertex, the measure w decreases by at least $\beta - \gamma$. We let $\Delta = \min\{1 - \alpha, \beta - \gamma\} = 0.041766$. Then the measure w decreases by at least $\gamma + 5\Delta$. In the second branch, a marked blue vertex v is deleted, and the measure w decreases by γ. We get a recurrence relation

$$C(w) \leq C(w - (\gamma + 5\Delta)) + C(w - \gamma), \tag{7}$$

the branching factor of which is not greater than 1.9967.

▶ **Lemma 7.** *If there is no marked vertex in B, after applying Reduction Rule 3, all marked vertices in the graph are in A.*

Proof. By Lemmas 3 and 4, we know that among all terminals, only t can possibly reach a marked vertex $v \in Y$, and any path from t to v only contains vertices in $A \cup Y$. In our algorithm, we never add new edges or new vertices. If a marked vertex v is not removed by Reduction Rule 3, then there is a path P from t to v that only contains vertices in $A \cup Y$. If v is not in A, then P must contain a vertex in B which is a marked vertex in Y. This finishes the proof. ◀

Lemma 7 says that when none of Steps 5 to 9 can be applied, there is no marked vertex in $B \cup R$. The vertices in A are not allowed to be selected to the solution set, and we can see that whether the vertices in A are marked or not does not affect our problem. When no marked vertex is in $B \cup R$, our algorithm will execute the first four steps.

For the sake of completeness, finally we prove the following property that is frequently used in the algorithm.

▶ **Lemma 8.** *The Basic Property of A always holds after each step following by applying reduction rules.*

Proof. For Step 4, we have proved this in Lemma 5. For other steps, deleting vertices may only violate item (2) of the Basic Property. Assume item (2) holds before executing an operation Π and after executing this operation Π terminal t can not reach a vertex $v \in A$. If Π deletes a vertex $u \in A$, then Π can only be Reduction Rule 3. For this case, vertex v will also be deleted by Reduction Rule 3 since u is on the path from t to v. If Π deletes a vertex $u \notin A$, then Π can only be one of Steps 5 to 9. For this case, the vertex u is a marked vertex in Y and only terminal t can reach v before deleting u by Lemma 3. After deleting u, no terminal can reach v and it will not be in any T-path and cycle containing a terminal. Thus, vertex v will also be deleted by Reduction Rule 3. ◀

We set α, β and γ as variables and generate a constraint for each of above branching operations to get a quasiconvex program. By solving this quasiconvex program according to the method introduced in [6], we get a bound of $O(1.9967^w)$ by setting $\alpha = 0.958234$, $\beta = 0.943555$ and $\gamma = 0.901789$ for our problem. Among all the branching recurrences in our algorithm, the worst ones are (3),(6), and (7), which have a branching factor not greater than 1.9967. Thus, Directed Multiway Cut can be solved in $O(1.9967^w)$ time. According to the setting of the vertex weight, we know that $w \leq n$. We get the following theorem.

▶ **Theorem 9.** *Directed Multiway Cut can be solved in $O(1.9967^n)$ time.*

4 Conclusion and Discussion

Fomin et al. [8] put forth a general method for a range of deletion problems to design exact algorithms and surpass the 2^n barrier by leveraging single-exponential parameterized algorithms. Given an $O^*(a^k)$-time parameterized algorithm where k is the solution size, we can derive an $O^*(b^n)$-time algorithm where $b = 2 - 1/a$. For Directed Multiway Cut, we have not found a single-exponential parameterized algorithm yet; for Directed Multicut, the problem even becomes W[1]-hard when there are at least four terminal pairs [18]. Consequently, the method in [8] cannot be currently applied to these two problems. Fortunately, we have managed to break 2^n for Directed Multiway Cut by proposing a relatively simple algorithm. It seems that directed problems are much harder than undirected problems. One of the most important techniques in this paper to achieve the breakthrough should be the technique of reversing the direction of edges. Before reversing the direction, we use some branching rules; after revising the direction, we use some other branching rules. This technique, not being used before as far as we know, makes the algorithm simple. The algorithm presented in this paper may aid us in understanding exhaustive-searching algorithms for related cut problems in directed graphs.

References

1 Gruia Călinescu, Cristina G. Fernandes, and Bruce A. Reed. Multicuts in unweighted graphs and digraphs with bounded degree and bounded tree-width. *J. Algorithms*, 48(2):333–359, 2003.

2 Jianer Chen, Yang Liu, and Songjian Lu. An improved parameterized algorithm for the minimum node multiway cut problem. *Algorithmica*, 55(1):1–13, 2009.

3 Rajesh Chitnis, Fedor V. Fomin, Daniel Lokshtanov, Pranabendu Misra, M. S. Ramanujan, and Saket Saurabh. Faster exact algorithms for some terminal set problems. *J. Comput. Syst. Sci.*, 88:195–207, 2017.

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

5 Elias Dahlhaus, David S. Johnson, Christos H. Papadimitriou, Paul D. Seymour, and Mihalis Yannakakis. The complexity of multiterminal cuts. *SIAM J. Comput.*, 23(4):864–894, 1994.

6 David Eppstein. Quasiconvex analysis of multivariate recurrence equations for backtracking algorithms. *ACM Trans. Algorithms*, 2(4):492–509, 2006. doi:10.1145/1198513.1198515.

7 Baris Can Esmer, Ariel Kulik, Dániel Marx, Daniel Neuen, and Roohani Sharma. Optimally repurposing existing algorithms to obtain exponential-time approximations. In David P. Woodruff, editor, *Proceedings of the 2024 ACM-SIAM Symposium on Discrete Algorithms, SODA 2024, Alexandria, VA, USA, January 7-10, 2024*, pages 314–345. SIAM, 2024. doi:10.1137/1.9781611977912.13.

8 Fedor V. Fomin, Serge Gaspers, Daniel Lokshtanov, and Saket Saurabh. Exact algorithms via monotone local search. *J. ACM*, 66(2):8:1–8:23, 2019.

9 Fedor V. Fomin, Serge Gaspers, Artem V. Pyatkin, and Igor Razgon. On the minimum feedback vertex set problem: Exact and enumeration algorithms. *Algorithmica*, 52(2):293–307, 2008.

10 Fedor V. Fomin, Fabrizio Grandoni, and Dieter Kratsch. A measure & conquer approach for the analysis of exact algorithms. *J. ACM*, 56(5):25:1–25:32, 2009. `doi:10.1145/1552285.1552286`.

11 Fedor V. Fomin, Pinar Heggernes, Dieter Kratsch, Charis Papadopoulos, and Yngve Villanger. Enumerating minimal subset feedback vertex sets. *Algorithmica*, 69(1):216–231, 2014.

12 Fedor V. Fomin and Dieter Kratsch. *Exact Exponential Algorithms*. Texts in Theoretical Computer Science. An EATCS Series. Springer, 2010.

13 Naveen Garg, Vijay V. Vazirani, and Mihalis Yannakakis. Multiway cuts in node weighted graphs. *J. Algorithms*, 50(1):49–61, 2004.

14 Russell Impagliazzo and Ramamohan Paturi. On the complexity of k-SAT. *J. Comput. Syst. Sci.*, 62(2):367–375, 2001.

15 Daniel Lokshtanov, Saket Saurabh, and Ondrej Suchý. Solving multicut faster than 2^n. In Andreas S. Schulz and Dorothea Wagner, editors, *Algorithms - ESA 2014 - 22th Annual European Symposium, Wroclaw, Poland, September 8-10, 2014. Proceedings*, volume 8737 of *Lecture Notes in Computer Science*, pages 666–676. Springer, 2014.

16 W. Mader. Über die maximalzahl kreuzungsfreier h-wege. *Arch. Math. (Basel)*, 31(4):387–402, 1978/1979.

17 Joseph Naor and Leonid Zosin. A 2-approximation algorithm for the directed multiway cut problem. *SIAM J. Comput.*, 31(2):477–482, 2001. `doi:10.1137/S009753979732147X`.

18 Marcin Pilipczuk and Magnus Wahlström. Directed multicut is w[1]-hard, even for four terminal pairs. *ACM Trans. Comput. Theory*, 10(3):13:1–13:18, 2018. `doi:10.1145/3201775`.

19 Igor Razgon. Exact computation of maximum induced forest. In Lars Arge and Rusins Freivalds, editors, *Algorithm Theory - SWAT 2006, 10th ScandinavianWorkshop on Algorithm Theory, Riga, Latvia, July 6-8, 2006, Proceedings*, volume 4059 of *Lecture Notes in Computer Science*, pages 160–171. Springer, 2006.

20 Mingyu Xiao. Simple and improved parameterized algorithms for multiterminal cuts. *Theory Comput. Syst.*, 46(4):723–736, 2010.

21 Mingyu Xiao and Hiroshi Nagamochi. An improved exact algorithm for undirected feedback vertex set. *J. Comb. Optim.*, 30(2):214–241, 2015.