Directed Multicut with linearly ordered terminals

Robert F. Erbacher
Army Research Lab
`robert.f.erbacher.civ@mail.mil`

Trent Jaeger
Penn State University
`tjaeger@cse.psu.edu`

Nirupama Talele
Penn State University
`nrt123@psu.edu`

Jason Teutsch
Penn State University
`teutsch@cse.psu.edu`

July 29, 2014

Abstract

Motivated by an application in network security, we investigate the following "linear" case of Directed Multicut. Let G be a directed graph which includes some distinguished vertices $t_1, \dots, t_k$. What is the size of the smallest edge cut which eliminates all paths from t_i to t_j for all $i < j$? We show that this problem is fixed-parameter tractable when parametrized in the cutset size p via an algorithm running in $O(4^p pn^4)$ time.

1 Multicut requests as partially ordered sets

The problem of finding a smallest edge cut separating vertices in a graph has received much attention over the past 50 years. Directed Multicut, one of the more general forms of this problem, encompasses numerous applications in algorithmic graph theory.

Name: Directed Multicut.

Instance: A directed graph G and pairs of terminal vertices $\{(s_1, t_1), \dots, (s_k, t_k)\}$ from G.

Problem: Find a smallest set of edges in G whose deletion eliminates all paths $s_i \to t_i$.

Special cases of the Directed Multicut problem have been met with success, although the general problem has no polynomial-time solution unless $\mathsf{P} = \mathsf{NP}$. The classical and efficient Ford-Fulkerson algorithm [9] solves Directed Multicut for the case of a single pair of terminal vertices, yet deciding whether there exists a minimum edge cut of a given size separating both s from t and t from s in a directed graph is NP-complete [10] as is deciding the size of a minimum edge cut separating three vertices in an undirected graph [8].

While Directed Multicut appears intractable from the perspective of NP-completeness, it remains an open problem to determine whether we can find an efficient parametrized solution for Directed Multicut. In practice we can optimize our solution based on other parameters besides the input length. In the case of Directed Multicut, the relevant parameters are the number of (sets of) terminal vertices k and and the size of the smallest solution, or *cutset*, p. Formally a problem is *fixed parameter tractable (FPT) in parameters k and p* if there exists an algorithm which, on input x, either gives a solution consistent with parameters k and p or correctly decides that no such solution exists in at most $f(k, p) \cdot \text{poly}(|x|)$ steps for some computable bound f.

Some subcases of Directed Multicut already have FPT solutions within the realm of fixed-parameter tractability. Recently Kratsch, Pilipczuk, Pilipczuk, and Wahlström [13] showed that Directed Multicut restricted to acyclic graphs is fixed-parameter tractable when parameterized in both the size of the cutset and the number of terminals. Chitnis, Hajiaghayi, and Marx [5], on the other hand, investigated Directed Multicut with restrictions of the terminal pairs. They showed that Directed Multiway Cut, the special case of Directed Multicut where all pairs of terminal vertices must be separated in both directions, is FPT when parametrized in just the size of the cutset. In the negative direction, Marx and Razgon [15] showed that Directed Multicut is W[1]-hard when parameterized the size of the cutset. Thus an FPT solution for Directed Multicut, if such an algorithm exists, most likely requires parameterization in the number of terminals in addition to the size of the cutset. We remark that in this same paper [15] Marx and Razgon also showed that the undirected Multicut problem is FPT when parametrized in the size of the cutset. Bousquet, Daligault, and Thomassé independently achieved this same result [2].

We now formalize the Poset Cut problem, a subject which derives from a network security framework [18]. We shall show that Poset Cut is equivalent to Directed Multicut with respect to fixed parameter tractability.

Name: Poset Cut

Instance: A directed graph $G = (V, E)$ with terminal vertices $T \subseteq V$, a partially ordered set P, and a surjective map $\ell : T \to P$.

Problem: Find a minimum set of edges $S \subseteq E$ so that for all terminal vertices $x, y \in T$, if there is a path from x to y in $(V, E \setminus S)$ then $\ell(x) \geq_P \ell(y)$.

The Poset Cut problem is immediately a special case of Directed Multicut. Indeed, given an instance of Poset Cut, we can read off from the poset P and mapping $\ell : T \to P$ those pairs of terminals which must be separated in the Poset Cut solution. These pairs together with the original input graph give us an instance of Directed Multicut such that an edge cut is a solution to the Poset Cut instance if and only if it is a solution to the Directed Multicut instance. Thus if Directed Multicut is fixed-parameter tractable, then so is Poset Cut. We now show that the reverse is also true.

Theorem 1. *If* Poset Cut *is* FPT*, then so is* Directed Multicut*. In particular, given an instance of* Directed Multicut *with k terminal pairs and a permitted maximum of p cuts, we can efficiently find an instance of* Poset Cut *with at most $2k$ terminal nodes and a permitted maximum of p cuts such that the* Poset Cut *instance has a solution iff the* Directed Multicut *instance does.*

Proof. Consider an instance of Directed Multicut consisting of a graph G, forbidden terminal pairs $s_1 \not\to t_1, \dots, s_k \not\to t_k$, and a cutsize parameter p. We define the corresponding Poset Cut instance as follows. The graph G' will consist of all the nodes and edges in G plus some extra nodes and edges. For each terminal node s_i, add a node a_i and enough paths from a_i to s_i so that a_i and s_i remain connected in any solution for the Poset Cut instance. In more detail

- add $p + 1$ nodes $c_{i,1}, \dots, c_{i,p+1}$,
- add an edge from a_i to each $c_{i,j}$, and
- add a further edge from each $c_{i,j}$ to s_i.

Similarly for each terminal node t_i, we add a node b_i and connect t_i to b_i with many paths: make $p + 1$ new nodes $d_{i,1}, \dots d_{i,p+1}$, add an edge from t_i to each $d_{i,j}$, and add an edge from each $d_{i,j}$ to b_i. We define the poset for this Poset Cut instance as follows: set a_i to be greater than b_j for all $i \neq j$, and all other pairs of terminal nodes are designated as incomparable.

By construction, there is a path $a_i \to b_i$ iff there is a path $s_i \to t_i$, and this condition holds even when up to p edges are deleted from G'. If there is a Poset Cut solution on G' under the given poset with at most p cuts, there is a further solution which is identical but avoids cutting any paths between a_i and s_i or t_i and b_i. Hence we may assume that the solution has all its cuts inside the embedding of G within G'. Transferring these cuts back to the original graph G gives a solution for the Directed Multicut instance. On the other hand, any solution for Directed Multicut in G will also be a solution for Poset Cut in G' because the only paths between pairs of terminal vertices in the Poset Cut instance start at some a_i and end at some b_j. □

Edwards, Jaeger, Muthukmaran, Rueda, Talele, Teutsch, Vijayakumar [16] and Jaeger, Teutsch, Talele, Erbacher [19] distilled the placement of host security mediators on a distributed system to a solution for the Poset Cut problem. They interpreted the components of a distributed system as nodes in a directed graph with edges indicating which components can communicate directly with others. Some information traveling through a network will have high integrity, and other information will have lower integrity, and security is achieved by blocking all flows from lower integrity to higher integrity nodes. Terminal nodes represent both the possible attack surfaces and higher integrity entities in the system, and each terminal corresponds to a specific integrity level as measured by the poset. In this context, we can interpret Poset Cut as a search for minimum intervention which mediates between all illegal information flows.

For the remainder of this paper, we will focus on the subcase of Poset Cut where the poset is a chain.

Name: Linear Cut

Instance: A directed graph (V, E) and a tuple of *terminal* sets $\langle T_1, \dots, T_k \rangle$ which are subsets of V.

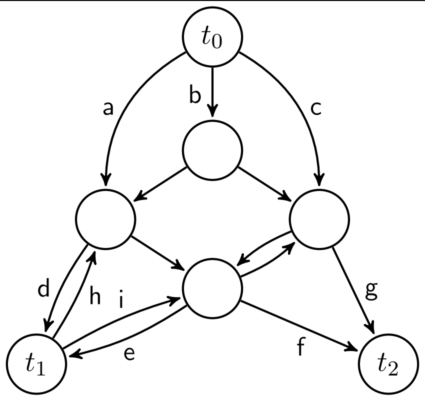
Problem: Find a smallest set of edges $S \subseteq E$ such that for any $s \in T_i$ and $t \in T_j$, if there is a path from s to t in $(V, E \setminus S)$, then $i \geq j$.

That is, Linear Cut wants to find a smallest edge cut which prevents every terminal set T_i from flowing to T_j whenever $j > i$. We shall show that Linear Cut, which is NP-hard in the sense of Proposition 6, is FPT when parameterized in the size of the cutset. Rephrased in terms of posets, Chitnis, Hajiaghayi, and Marx's algorithm [5] for Directed Multiway Cut shows that Poset Cut is FPT parametrized in the cutset size when the underlying poset is an antichain.

2 A parameterized algorithm for Linear Cut

We shall show that Linear Cut is FPT when parametrized in the size of cutset. Before presenting our parametrized algorithm, we first analyze the following example which illustrates why the naïve greedy cut does not yield an optimal solution. The graph given in Figure 1 has three terminal vertices t_0, t_1, and t_2, and we would like to find a small set of edges whose removal eliminates all paths from t_0 to either t_1 or t_2 as well as all paths from t_1 to t_2. Consider the greedy algorithm which uses the Ford-Fulkerson algorithm to first eliminate all paths from t_0 to the other terminal vertices and then again to extinguish the paths from t_1 to t_2. A minimal edge cut from t_0 to the set $\{t_1, t_2\}$ has size 3, so let us assume that the algorithm chooses edges $\{a, b, c\}$. Now a minimal edge cut from t_1 to t_2 has size 2, for example $\{h, i\}$. Thus this greedy algorithm solves the Linear Cut instance with a cut of size 5. On the other hand, $\{d, e, f, g\}$ is a solution of size 4.

Figure 1 The greedy algorithm is not optimal.



We now describe our parametrized solution for Linear Cut. Our algorithm either outputs a solution cut of size less $\leq p$ or returns NO if no such cut exists. Our construction exploits a technique used in Chen, Liu, and Lu's fixed-parameter solution [3] to the Multiway Cut problem in undirected graphs which improved a result of Marx [14]. A similar idea appeared earlier in Chen, Liu, Lu, O'Sullivan, and Razgon's algorithm [4] for Skew Separator, a key step in their parametrized solution for Directed Feedback Vertex Set. We remark that the pushing of important separators technique along the lines of [14, Theorem 3.7] gives a parameterized solution for Linear Cut in time $O(4^{p^3} n^{O(1)})$, and using a reduction to the Skew Separator algorithm in [4] one can also show that Linear Cut has a solution which runs in the same time as the algorithm given below, namely $O(4^p pn^4)$.

An (X,Y)-*separator* is a set of edges such that any path from X to Y passes through one of its members. Our solution, Algorithm 1 proceeds in two phases. First we handle the trivial cases where $\mathcal{T} = \langle\rangle$, $p = 0$, or T_1 is either already separated from the other terminals or can't be separated with p edge cuts (lines 1–12). The second phase picks an edge pointing out of the T_1 region and checks whether making it undeleteable hurts the min size of a $(T_1, T_2 \cup \cdots \cup T_k)$-separator. If not we add the edge to the list of undeleteable edges, and if so we branch on the only two possibilities: either the edge belongs in the Linear Cut solution or it doesn't.

The following theorem gives the main justification for this algorithm. A set of edges is a *linear cut* with respect to the k-tuple of terminals $\langle T_1, \ldots, T_k\rangle$ if there is no path from T_i to T_j whenever $i < j$ once these edges have been removed.

Algorithm 1 FPT algorithm for LINEAR CUT parameterized in cutset size.

Input: A graph $G = (V, E)$, a k-tuple of terminal sets $\langle T_1, T_2, \dots T_k \rangle$ which are subsets of V, some undeletable edges $F \subseteq E$, and a parameter p.

Output: A set of $\leq p$ edges in $E \setminus F$ such that when these edges are deleted from G there is no path from T_i to T_j for any $i < j$, if such a set of edges exists, otherwise return NO.

```
1: function LC$((V, E), \langle T_1, \dots, T_k \rangle, F, p)$
2:     For ease of reading, let $\mathcal{T} = \langle T_1, \dots, T_k \rangle$.
3:     if $\mathcal{T} = \langle\rangle$ then return $\emptyset$;
4:     else if $p \leq 0$ then
5:         if for all $i < j$, $T_j$ is not reachable from $T_i$ in $G$ then return $\emptyset$;
6:         else return NO;
7:         end if
8:     end if
9:     let $m$ be the size of a minimum $(T_1, T_2 \cup \dots \cup T_k)$-separator which does not
   include edges from $F$.
10:    if $m > p$ or no separator exists due to undeleteable edges then return NO;
11:    else if $m = 0$ then return LC$((V, E), \langle T_2, \dots, T_k \rangle, F, p)$;
12:    else
13:        let $e \in E \setminus F$ be an edge with a tail reachable from $T_1$ via undeleteable
   edges.
14:        if the size of a minimum $(T_1, T_2 \cup \dots \cup T_k)$-separator which does not
   include edges from $F \cup \{e\}$ exists and is equal to $m$, then
15:            return LC$((V, E), \mathcal{T}, F \cup \{e\}, p)$;
16:        else if $\{e\} \cup$ LC$((V, E \setminus \{e\}), \mathcal{T}, F, p - 1)$ or LC$((V, E), \mathcal{T}, F \cup \{e\}, p)$ is
   not NO, then
17:            return the first of these two found to have a solution;
18:        else
19:            return NO;
20:        end if
21:    end if
22: end function
```

Theorem 2. *Let $\langle (V, E), \mathcal{T}, F, p\rangle$ be an input to Algorithm 1, where $\mathcal{T}$ is an abbreviation for $\langle T_1, \ldots, T_k\rangle$, and let e be an edge pointing from some node reachable from T_1 via undeleteable edges to a node outside $T_1 \cup F$. Suppose that the smallest $(T_1, T_2 \cup \cdots \cup T_k)$-separator with undeletable edges F is the same size as the smallest $(T_1, T_2 \cup \cdots \cup T_k)$-separator with undeletable edges $F \cup \{e\}$ and has cardinality at most p. Then the smallest linear cut among the terminal sets $\langle T_1, \ldots, T_k\rangle$ with undeletable edges F in (V, E) has the same size as the smallest linear cut among these same terminals with undeletable edges $F \cup \{e\}$.*

Proof. First note that making edges undeleteable can only increase the size of the smallest cut. Hence it suffices to show, under the hypothesis of the theorem, that the smallest linear cut with forbidden edges $F \cup \{e\}$ is no bigger than a minimal linear cut with forbidden edges F.

Let S be a minimal $(T_1, T_2 \cup \cdots \cup T_k)$-separator with undeletable edges $F \cup \{e\}$. Then S is also a separator between these same sets with undeletable edges F, and by the assumption of the theorem S is also a minimal such separator. Let W be a minimal linear cut in $G = (V, E)$ for $\mathcal{T}$ with undeletable edges F, and let R denote the set of edges that are reachable from T_1 in $(V, E \setminus S)$. We shall show that $W' = (W \cup S) \setminus R$ is a linear cut in G for $\mathcal{T}$ with undeletable edges $F \cup \{e\}$ which is no larger than W. Since making edges undeletable can only increase the size of a smallest solution, W' will indeed be minimal.

For clarity, we reformulate the problem instance without undeletable edges. We replace each undeletable edge $(x, y) \in F \cup \{e\}$ with $p + 1$ new, regular edges from x to y, whereby transforming the graph into a multigraph without any undeletable edges. Now any linear cut (resp. $(T_1, T_2 \cup \cdots \cup T_k)$-separator) consisting of at most p edges will be a solution in the transformed multigraph if and only if it is a solution in the original graph. The reason is that there are not enough total cuts in the instance to sever connectivity between any vertices with $p + 1$ multiedges. Thus these edges are effectively undeletable, and of course cuts not involving undeletable edges or multiedges will work the same in both the original and transformed instance.

First we argue that W' is not larger than W by proving $|S \setminus W| \le |W \cap R|$. Since S does not contain any of the undeletable, multiedge parts of G, by Menger's Theorem [12, Theorem 7.45], or more precisely its generalization to sets of vertices [3, Lemma 1], there are $|S|$ disjoint edge paths from T_1 to $\bigcup_{j>1} T_j$, each containing an edge in S. It follows that there are $|S \setminus W|$ disjoint edge paths from T_1 to $S \setminus W$. Now suppose that $|W \cap R| < |S \setminus W|$. Then there must be a path from T_1 to some edge $x \in S \setminus W$ which avoids $W \cap R$. Furthermore, by minimality of S, there is a path from x to some terminal set T_j with $j > 1$. But now there is a path from T_1 to some T_j which avoids W, contradicting that W is a linear cut.

It remains to show that W' is in fact a linear cut in G for $\mathcal{T}$ with undeletable edges $F \cup \{e\}$. Let Q be a forbidden path. If Q does not intersect R, then it must pass through $W \setminus R$ and hence through W'. On the other hand, suppose that Q does pass through R. Since T_1 is the least-indexed terminal set, Q must end at T_j for some $j > 1$, and therefore Q must pass through $S \subseteq W'$. In either case, removing W' eliminates the forbidden path Q. □

Theorem 3. *Algorithm 1 finds a solution in time $O[4^p p \cdot (|V| + |E|) \cdot |E|]$, if one exists, and outputs NO otherwise.*

Proof. Line 13 of Algorithm 1 selects an edge $e \in E \setminus F$ for consideration. If the condition for edge e in line 14 holds, then preserving e does not hurt the $(T_1, T_2 \cup \cdots \cup T_k)$-separator, and therefore by Theorem 2 no harm comes to the Linear Cut instance by adding e to the list of undeletable edges. If this condition is not satisfied, then the algorithm exhaustively searches both for a solution containing the edge e (Option 1) and for a solution not containing e (Option 2). In Option 1, the algorithm searches for a solution of size $p - 1$ containing e, and in Option 2, the size of the smallest$(T_1, T_2 \cup \cdots \cup T_k)$-separator increases by 1. Along any branch of the algorithm, either of these two Options can occur at most p times for each terminal before the algorithm returns NO, and the latter happens only when exhaustive search fails to find a solution. Hence the algorithm eventually terminates with the correct answer.

We can refine our analysis further to show that there are at most 4^p possible branches in the algorithm. We argue that any branch of the algorithm witnesses at most $2p$ branching splits. Suppose that the initial input parameter is p and that the smallest $(T_1, T_2 \cup \cdots \cup T_k)$-separator has size m. Since each iteration of Option 1 decreases the size of the minimal $(T_1, T_2 \cup \cdots \cup T_k)$-separator by 1, the path which always chooses Option 1 will witness exactly m branches up to the point where Line 11 of Algorithm 1 recognizes that T_1 has been separated and removes it from further consideration. Each time Option 2 is chosen along the path, the size of the smallest$(T_1, T_2 \cup \cdots \cup T_k)$-separator increases by at least 1, so if Option 2 happens r times, then Option 1 must happen a total of at least $m + r$ times before T_1 is separated. Thus the size of the cutset size parameter when T_1 becomes separated is at most $p - m - r$, the initial parameter value minus the number of times Option 1 was chosen, and the total number of splits witnessed is $(m + r) + r$, which is at most twice the number of edges added to the cutset. The same counting argument holds for separators for successive T_i's and it follows that each search path can witness at most $2p$ splits in case the algorithm succeeds.

The number of steps between each encounter with an Option is essentially the time required to check whether a separator size p exists, which is $O[p(|V| + |E|)]$ by the argument in [3, Lemma 2], times the number edges. The multiplicative factor of $|E|$ comes from the potential recursion in line 15. Hence the total runtime is $O[2^{2p} p \cdot (|V| + |E|) \cdot |E|]$. □

Corollary 4. Linear Cut *is fixed-parameter tractable when parameterized in the size of the cutset.*

3 Hardness result

Marx and Razgon [15] showed that Directed Multicut parameterized in the size of the cutset is W[1]-hard by reducing this problem to the known W[1]-hard problem Clique. Therefore the following is immediate from Theorem 1.

Corollary 5. Poset Cut *is* W[1]*-hard when parameterized in the size of the cutset.*

Whether Directed Multicut is fixed-parameter tractable when parameterized in both the size of the cutset and the number of terminals remains an open problem, even in the case where we fix the number of terminal pairs at $k = 3$ [5, 15]. Linear Cut for $k = 2$ is possible via the Ford-Fulkerson algorithm, however for longer chains the problem also becomes NP-hard.

Proposition 6. *Deciding whether a* Linear Cut *instance has a solution of size* p *is* NP*-complete for* $k = 3$ *terminals.*

Proof. Linear Cut is trivially in NP as one can easily check by breadth-first search whether a given set of edges is a solution.

We reduce the undirected Multiway Cut problem for $k = 3$, which is NP-hard [8], to the Linear Cut problem for $k = 3$. Let G be an undirected graph with terminal nodes s, t and u be an instance of Multiway Cut, the problem of finding a smallest edge cut which separates s, t, and u. Construct a new directed graph G' which has the same vertices as G except for each edge $e = \{x, y\}$ in G we also add two new vertices a_e and b_e. The edges from G do not carry over to G', and instead we add directed edges (x, a_e), (y, a_e), (a_e, b_e), (b_e, y), and (b_e, x). We call this collection of edges the *gadget* for e. Our Linear Cut instance consists of the graph G' together with the embedded terminals nodes s, t, and u from G with the (arbitrary) tuple ordering $\langle s, t, u\rangle$. Technically we treat the terminal nodes here as singleton sets when formulating this instance of Linear Cut.

Assume $C = \{e_1, \dots, e_p\}$ is a Multiway Cut solution for G. We claim that $C' = \{(a_{e_1}, b_{e_1}), \dots, (a_{e_p}, b_{e_p})\}$ is then a Linear Cut solution for G'. Suppose there were some prohibited path in G' between two terminals, say s and t, which avoids C'. This path must have the form

$$s \to a_{(s,x_1)} \to b_{(s,x_1)} \to x_1 \to a_{(x_1,x_2)} \to b_{(x_1,x_2)} \to x_2 \to \cdots \to t$$

for some vertices $x_1, x_2, \dots$ in G. Contracting all the a_i's and b_i's from this path yields a path from s to t in G which avoids C, which is impossible.

Conversely, assume that $C' = \{d_1, \dots, d_p\}$ is a Linear Cut solution for G. For each $i \leq p$, let e_i be the gadget for the edge in G which d_i belongs to. Then $C = \{e_1, \dots, e_p\}$ is a Multiway Cut solution for G as any path $x_1 \to \cdots \to x_k$ between terminals in G avoiding C gives rise to a path between the same terminals in G' which avoids C', namely

$$x_1 \to a_{(x_1,x_2)} \to b_{(x_1,x_2)} \to x_2 \to a_{(x_2,x_3)} \to b_{(x_2,x_3)} \to x_3 \to \cdots \to x_k,$$

which cannot exist. Thus Multiway Cut is polynomial-time reducible to Linear Cut. ◻

4 Approximation

It seems difficult to efficiently approximate Directed Multicut [1, 6, 11], which indicates that Poset Cut may not have a good approximation algorithm either. The best known polynomial-time approximation algorithm for Directed Multicut is just under $O(\sqrt{n})$ [1]. We wonder whether Linear Cut may be easier to approximate.

Recall that Directed Multiway Cut is the problem of Poset Cut restricted to the instances where the underlying poset is an antichain.

Name: Directed Multiway Cut

Instance: A directed graph (V, E) and a tuple of *terminal* sets $T_1, \ldots, T_k$ which are subsets of V.

Problem: Find a smallest set of edges $S \subseteq E$ such that there is no path from T_i to T_j in $(V, E \setminus S)$ for all $i \neq j$.

Garg, Vazirani, and Yannakakis [10] gave a $2 \log n$ approximation for Directed Multiway Cut, later improved to a factor of 2 by Naor and Zosin [17] using an LP relaxation. The undirected Multiway Cut problem for k terminals has a simple $2 - 2/k$ approximation algorithm using isolated cuts [8] and even a $1.5 - 2/k$ approximation using LP relaxation [7] (see also [20]). By making two calls to Algorithm 1, we can obtain a simple approximation to Directed Multiway Cut which runs faster than Chitnis, Hajiaghayi, and Marx's $2^{2^{O(p)}} n^{O(1)}$-time exact solution [5] but does not beat Naor and Zosin's polynomial-time 2-approximation [17].

Corollary 7. *One can find a solution for* Directed Multiway Cut *of instance size n in time $O(4^p p n^4)$ which is within a factor of two of optimal whenever a solution of size p exists.*

Proof. Assume that $T_1, \ldots, T_k$ are the terminal sets which need to be separated in the directed graph (V, E). Using Algorithm 1, make one Linear Cut which cuts using the terminal sets $\langle T_1, \ldots, T_k \rangle$ and another which uses this k-tuple reversed, $\langle T_k, \ldots, T_1 \rangle$. The union of these two cuts is a solution to the Directed Multiway Cut instance, when both exist, and neither cut is larger than the smallest possible solution. ◻

References

[1] Amit Agarwal, Noga Alon, and Moses S. Charikar. Improved approximation for directed cut problems. In *Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*, STOC '07, pages 671–680, New York, NY, USA, 2007. ACM.

[2] Nicolas Bousquet, Jean Daligault, and Stéphan Thomassé. Multicut is fpt. In *Proceedings of the 43rd annual ACM symposium on Theory of computing*, STOC '11, pages 459–468, New York, NY, USA, 2011. ACM.

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

[4] Jianer Chen, Yang Liu, Songjian Lu, Barry O'Sullivan, and Igor Razgon. A fixed-parameter algorithm for the directed feedback vertex set problem. *Journal of the ACM*, 55(5):21:1–21:19, November 2008.

[5] Rajesh Chitnis, MohammadTaghi Hajiaghayi, and Dániel Marx. Fixed-parameter tractability of directed multiway cut parameterized by the size of the cutset. In *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '12, pages 1713–1725. SIAM, 2012.

[6] Julia Chuzhoy and Sanjeev Khanna. Hardness of cut problems in directed graphs. In *Proceedings of the thirty-eighth annual ACM symposium on Theory of computing*, STOC '06, pages 527–536, New York, NY, USA, 2006. ACM.

[7] Gruia Călinescu, Howard Karloff, and Yuval Rabani. An improved approximation algorithm for multiway cut. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, STOC '98, pages 48–52, New York, NY, USA, 1998. ACM.

[8] E. Dahlhaus, D. S. Johnson, C. H. Papadimitriou, P. D. Seymour, and M. Yannakakis. The complexity of multiway cuts (extended abstract). In *Proceedings of the twenty-fourth annual ACM symposium on Theory of computing*, STOC '92, pages 241–251, New York, NY, USA, 1992. ACM.

[9] L. R. Ford and D. R. Fulkerson. Maximal flow through a network. Technical report, 1956.

[10] Naveen Garg, Vijay Vazirani, and Mihalis Yannakakis. Multiway cuts in directed and node weighted graphs. In Serge Abiteboul and Eli Shamir, editors, *Automata, Languages and Programming*, volume 820 of *Lecture Notes in Computer Science*, pages 487–498. Springer Berlin / Heidelberg, 1994.

[11] Anupam Gupta. Improved results for directed multicut. In *Proceedings of the fourteenth annual ACM-SIAM symposium on Discrete algorithms*, SODA '03, pages 454–455, Philadelphia, PA, USA, 2003. Society for Industrial and Applied Mathematics.

[12] Jon Kleinberg and Eva Tardos. *Algorithm Design*. Addison-Wesley, 2005.

[13] Stefan Kratsch, Marcin Pilipczuk, Michal Pilipczuk, and Magnus Wahlström. Fixed-parameter tractability of multicut in directed acyclic graphs. `http://arxiv.org/abs/1202.5749`.

[14] Dániel Marx. Parameterized graph separation problems. *Theoretical Computer Science*, 351(3):394–406, 2006.

[15] Dániel Marx and Igor Razgon. Fixed-parameter tractability of multicut parameterized by the size of the cutset. In *Proceedings of the 43rd annual ACM symposium on Theory of computing*, STOC '11, pages 469–478, New York, NY, USA, 2011. ACM.

[16] Divya Muthukumaran, Sandra Rueda, Nirupama Talele, Hayawardh Vijayakumar, Trent Jaeger, Jason Teutsch, and Nigel Edwards. Transforming commodity security policies to enforce Clark-Wilson integrity. In *Proceedings of the 28th Annual Computer Security Applications Conference (ACSAC 2012)*, December 2012.

[17] J. Naor and L. Zosin. A 2-approximation algorithm for the directed multiway cut problem. In *Proceedings of the 38th Annual Symposium on Foundations of Computer Science*, (FOCS '97), pages 548–553, 1997.

[18] Lee Pike. Post-hoc separation policy analysis with graph algorithms. In *Workshop on Foundations of Computer Security (FCS 2009). Affiliated with Logic in Computer Science (LICS)(August 2009)*, 2009.

[19] Nirupama Talele, Jason Teutsch, Trent Jaeger, and Robert F. Erbacher. Using security policies to automate placement of network intrusion prevention. In *Engineering Secure Software and Systems*, volume 7781 of *Lecture Notes in Computer Science*, pages 17–32. Springer, Berlin Heidelberg, 2013.

[20] Vijay V. Vazirani. *Approximation algorithms*. Springer-Verlag, Berlin, 2003.