On the complexity of the multicut problem in bounded tree-width graphs and digraphs

Cédric Bentz*

Abstract

Given an edge- or vertex-weighted graph or digraph and a list of source-sink pairs, the *minimum multicut problem* consists in selecting a minimum weight set of edges or vertices whose removal leaves no path from each source to the corresponding sink. This is a classical $\mathcal{NP}$-hard problem, and we show that the edge version becomes tractable in bounded tree-width graphs if the number of source-sink pairs is fixed, but remains $\mathcal{NP}$-hard in directed acyclic graphs and $\mathcal{APX}$-hard in bounded tree-width and bounded degree unweighted digraphs. The vertex version, although tractable in trees, is proved to be $\mathcal{NP}$-hard in unweighted cacti of bounded degree and bounded path-width.

Keywords: multicuts, $\mathcal{NP}$-hardness, $\mathcal{APX}$-hardness, bounded tree-width.

1 Introduction

In this paper, we study the complexity of the well-known *minimum multicut problem* (MinMC). This problem is one of the fundamental problems in graph theory, and has many applications [4].

Assume we are given a n-vertex m-edge (di)graph $G = (V, E)$, a *weight function* $c : E \rightarrow \mathbb{N}^*$ and a list $\mathcal{N}$ of pairs (source s_i, sink s'_i) of *terminal* vertices. Each pair (s_i, s'_i) is a *net*. MinMC consists in selecting a minimum weight set of edges (or arcs) whose removal *disconnects* each one of the nets (s_i, s'_i), i.e., leaves no (directed) path from s_i to s'_i for each i. The *minimum multiterminal cut problem* (MinMTC) is a particular minimum multicut problem in which, given a set of ρ vertices $\{t_1, \ldots, t_\rho\}$, the nets are (t_i, t_j) for $i \neq j$. Note that, for both problems, there exists a variant where we want to remove vertices instead of edges (in this case, the weight function is defined on the vertices, i.e., $c : V \rightarrow \mathbb{N}^*$), and that the graph can be directed or undirected. Moreover, when removing vertices, we can allow to remove

*LRI, Université Paris-Sud and CNRS, 91405 Orsay Cedex, France (e-mail: cedric.bentz@lri.fr)

terminal vertices or not. We use the letters "V" and "E" to denote the vertex and edge variants respectively, the letter "D" for the directed variant, and the letter "U" (for *unrestricted*, as in [1]) when we allow to remove terminal vertices. For instance, UVMinMC and DEMinMTC respectively denote the unrestricted vertex (undirected) variant of MinMC and the directed edge variant of MinMTC.

For $|\mathcal{N}| = 1$, EMinMC is equivalent to the *minimum cut problem*, and therefore is polynomial-time solvable both in directed and in undirected graphs. However, MinMC and MinMTC become $\mathcal{NP}$-hard (and even $\mathcal{APX}$-hard, see below for a definition) as soon as $|\mathcal{N}| = 3$ both in undirected and in directed graphs [5], and, for an arbitrary number of nets, EMinMC is $\mathcal{NP}$-hard even in unweighted stars [10]. Moreover, DEMinMC is polynomial-time solvable in directed trees (the constraint matrix being totally unimodular) and DEMinMTC is polynomial-time solvable in directed acyclic graphs [4]. Eventually, there is a polynomial-time approximation scheme (PTAS) for EMinMC in unweighted graphs of bounded tree-width and bounded degree, but dropping any of these three assumptions leads to $\mathcal{APX}$-hardness (instead of $\mathcal{NP}$-hardness only) [1]. This PTAS also holds for a variant of DEMinMC, called DEMinMC-SC, where the goal is to remove a minimum weight set of arcs so that, for each i, no directed cycle contains both s_i and s'_i. Recall that a minimization problem is $\mathcal{APX}$-hard if there exists a real $\epsilon > 0$ such that computing an $(1 + \epsilon)$-approximation for all instances of this problem is $\mathcal{NP}$-hard, where, for a real $\alpha > 1$, an α-approximation is a feasible solution whose value is at most α times the optimal value. The main approximation results for MinMC are that it can be approximated in polynomial time within $O(\log n)$ in undirected graphs [9] (within 2 in undirected trees [10]) and within $O(\sqrt{n})$ in digraphs [12], while for DEMinMC-SC an $O(\log^2 |\mathcal{N}|)$-approximation algorithm is known [15]. See [4] for further results and references concerning MinMC and MinMTC.

The vertex variants are studied in [1], where it is proved that UVMinMC is polynomial-time solvable in unweighted trees and admits a PTAS in bounded tree-width unweighted graphs (actually, this PTAS is the basis of the PTAS for EMinMC). Moreover, UVMinMC is $\mathcal{NP}$-hard in series-parallel graphs (i.e., in graphs of tree-width 2) with maximum degree 3, and VMinMC is $\mathcal{NP}$-hard in unweighted trees with maximum degree 4.

We show that EMinMC is tractable in bounded tree-width graphs when $|\mathcal{N}|$ is fixed and that, although UVMinMC is known to be tractable in trees, it is $\mathcal{NP}$-hard in unweighted cacti of bounded degree and bounded path-width (Sect. 3). Recall that a *cactus* is a graph where any edge lies on at most one cycle. We also prove that, unlike DEMinMTC, DEMinMC is $\mathcal{NP}$-hard in directed acyclic graphs, even if the graph is restricted to be an unweighted cactus of bounded in- and out- degrees (Sect. 4). Eventually, we show that DEMinMC is $\mathcal{APX}$-hard in bounded tree-width and bounded degree unweighted digraphs (Sect. 4). We start with some useful definitions.

2 Preliminaries

2.1 Definitions

In this section, we recall the definition of the *tree-width* of a graph. We begin with the undirected case (see [17]):

Definition 1. *Let $G = (V, E)$ be an undirected graph. We say that a pair $\Theta = (T, (X_w)_{w \text{ is a vertex of } T})$, consisting of a tree T and of a multiset whose elements $X_w \subseteq V$ (called* bags*) are indexed by the vertices of T, is a* tree decomposition *of G if it satisfies:*

- $\forall (u, v) \in E$, *there is a vertex w of T such that $u \in X_w$ and $v \in X_w$;*
- $\forall v \in V$, *the vertices $\{w : v \in X_w\}$ induce a connected subgraph of T.*

The width *of the tree decomposition Θ is equal to $\max_{w \text{ is a vertex of } T} |X_w| - 1$ and the* tree-width *of G, denoted by $tw(G)$, is the minimum of the widths of all tree decompositions of G.*

The trees are the graphs of tree-width 1. The *path-width* of a graph G is defined as the smallest integer δ such that G admits a tree decomposition of width δ where the tree T is a path.

We now define the tree-width notion for digraphs, which differs from the one used for undirected graphs. As in [1], we use the definition given in [14].

Definition 2. *An* arboreal decomposition of a digraph $D = (V, A)$ *is a triple (R, X, W), where $R = (U, F)$ is a rooted tree, $W = (W_r \subseteq V : r \in U)$ is a partition of V, and $X = (X_e \subseteq V : e \in F)$ satisfies:* if $e = (a, b) \in F$, then $Z_e = \bigcup \{W_r : r \in U$ is contained in the subtree of R rooted at $a\}$ is such that any directed path (elementary or not) in $D - X_e$ which starts and ends in Z_e is entirely contained in Z_e. *The* width of (R, X, W) *is the smallest integer δ such that $\left| \left(\bigcup_{e \text{ is adjacent to } r} X_e \right) \cup W_r \right| \leq \delta + 1$ for all r in U. The* tree-width of D *is the smallest integer δ such that D has an arboreal decomposition of width δ.*

The directed acyclic graphs are the digraphs of tree-width 0. Moreover, given an undirected graph G of tree-width δ, the digraph obtained from G by replacing every edge by two opposite arcs has directed tree-width δ.

2.2 A general scheme to prove the $\mathcal{NP}$-hardness of MinMC

In [1], Călinescu et al. prove that EMinMC is $\mathcal{NP}$-hard in unweighted binary (undirected) trees. Although this result is interesting enough in itself, its main interest lies in its proof. Indeed, the basic idea of this proof is quite powerful, in the sense that it can easily be adapted to prove the $\mathcal{NP}$-hardness of MinMC in various special cases (for both edge and vertex variants). Since

we will use this general idea in several of our own proofs, we detail in this section the main properties that we shall need.

The reduction is from the $\mathcal{NP}$-complete problem 3SAT [8], and we associate a gadget (called "variable gadget") to each one of the ν variables x_i and a gadget (called "clause gadget") to each one of the μ clauses C_j. All these gadgets are vertex-disjoint, unweighted and linked together via a common path Q: each gadget is linked to a vertex of Q, and no two gadgets are linked to the same vertex of Q. No terminal lies on Q, and a net is an *intra-gadget net* if its two endpoints lie in the same gadget, an *inter-gadget net* otherwise. The variable and clause gadgets have the following properties:

1. The i^{th} variable gadget has two vertices labelled x_i and $\bar{x}_i$. There is a (intra-gadget) net between two vertices of this gadget and there are two (mutually exclusive) ways of disconnecting this net by removing one edge/vertex: in the first case, there remains a path from the vertex labelled x_i (for the sake of simplicity, we shall just say *from* x_i) to the vertices of Q; in the second case, there remains a path from (the vertex labelled) $\bar{x}_i$ to the vertices of Q;

2. The gadget of clause $C_j = \hat{x}_a \vee \hat{x}_b \vee \hat{x}_c$ (where $\hat{x}_i$ is x_i or $\bar{x}_i$ for each i) has three particular vertices, labelled $\hat{x}_a$, $\hat{x}_b$ and $\hat{x}_c$. There are two (intra-gadget) nets between vertices of this gadget, and at least two edges/vertices are needed in order to disconnect them (one for each net). There are four (mutually exclusive) ways of disconnecting these two nets by removing two edges/vertices: in the two first cases, there remains a path from the vertices of Q to $\hat{x}_a$; in the third case, there remains a path from the vertices of Q to $\hat{x}_b$, and in the fourth case there remains a path from the vertices of Q to $\hat{x}_c$;

3. For each i, there is a (inter-gadget) net between the vertex labelled x_i (resp. labelled $\bar{x}_i$) in the i^{th} variable gadget and every vertex labelled x_i (resp. labelled $\bar{x}_i$) in a clause gadget.

The intra-gadget nets guarantee that at least $\nu + 2\mu$ edges/vertices are required in any feasible multicut (one in each variable gadget and two in each clause gadget). The inter-gadget nets guarantee that $\nu + 2\mu$ edges/vertices are enough if and only if there is a satisfying truth assignment (STA, for short) for the corresponding instance of 3SAT. Indeed, if a STA exists, we can construct an edge/vertex multicut of size $\nu + 2\mu$ as follows:

- In the gadget of each variable x_i, we remove (i.e., select in the multicut) an edge/a vertex in such a way that there remains a path from x_i to the vertices of Q if and only if x_i is *false* in the STA;

- In the gadget of each clause C_j, we remove two edges/vertices in such a way that there remains a path from the vertices of Q to a vertex labelled by a literal *true* in the STA.

Conversely, if there exists a multicut of size $\nu + 2\mu$, then, necessarily, there is one removed edge/vertex in each variable gadget and two in each clause gadget. For each i, we set the variable x_i to *true* if there remains a path from $\bar{x}_i$ to the vertices of Q, and to *false* otherwise. This way, we obtain a STA. Indeed, given a clause gadget, one of the three vertices labelled by literals in the gadget is such that there remains a path from the vertices of Q to itself. Thus, if this literal is x_j (resp. $\bar{x}_j$) for some j, then in the jth variable gadget there must remain no path from x_j (resp. $\bar{x}_j$) to the vertices of Q. Hence x_j is *true* (resp. *false*) in the STA, and the clause is satisfied.

This implies the $\mathcal{NP}$-hardness of MinMC whenever we can build variable and clause gadgets satisfying the required properties.

Remark 1. *There may actually be additional ways of disconnecting the intra-gadget net (resp. nets) of a variable (resp. clause) gadget by removing one edge/vertex (resp. two edges/vertices). These ways are different from the "mutually exclusive ways" (which are needed anyway), and are not necessarily mutually exclusive. However, these other ways must be such that:*

- *in the gadget of variable x_i, there remains a path from at least one of the two vertices x_i and $\bar{x}_i$ to the vertices of Q;*
- *in the gadget of clause $C_j = \hat{x}_a \vee \hat{x}_b \vee \hat{x}_c$, there remains a path from the vertices of Q to at least one of the three vertices $\hat{x}_a$, $\hat{x}_b$ and $\hat{x}_c$.*

3 Undirected Graphs

In this section, we first prove that, for fixed $|\mathcal{N}|$, EMinMC is polynomial-time solvable in bounded tree-width graphs, while dropping any of the two assumptions leads to $\mathcal{NP}$-hardness and even to $\mathcal{APX}$-hardness ([5], [10]). We use as a basic step the following idea from [5]: EMinMC can be solved by solving at most $\frac{\left(\sqrt{2|\mathcal{N}|}+1\right)^{2|\mathcal{N}|}}{\left(\sqrt{2|\mathcal{N}|}+1\right)!}$ instances of EMinMTC. Indeed, in any optimal solution for an instance of EMinMC, the $2|\mathcal{N}|$ terminals are clustered in $q \leq \sqrt{2|\mathcal{N}|}+1$ sets (or *clusters*), such that s_i and s'_i do not belong to the same cluster for each i and, for each pair of clusters, there is an i such that s_i is in one cluster and s'_i is in the other (otherwise, we can merge the two clusters and still have a valid clustering). Given a clustering, for each cluster, we add one new vertex (called a *cluster vertex*) linked by a new edge (called a *cluster edge* and having a sufficiently large weight) to every terminal contained in this cluster in order to transform the instance I of EMinMC into an instance I' of EMinMTC (where the terminals are the cluster vertices). Clearly, the optimal values of I' and I are equal, and any optimal solution to I' is an optimal solution to I. It remains to solve I' in polynomial time: in the general case, this is also an $\mathcal{NP}$-hard problem (the

authors of [5] introduced this transformation for approximation purposes). When the graph has bounded tree-width, we show that this is easier.

Theorem 1. *If $|\mathcal{N}|$ is fixed,* EMinMC *is polynomial-time solvable in graphs with bounded tree-width.*

Proof. We use the above transformation and the fact that, according to [5], EMinMTC is polynomial-time solvable in graphs with bounded tree-width, by standard dynamic programming techniques. Before adding the cluster vertices and edges, the graph has bounded tree-width by assumption, but we still have to prove that this remains true after they have been added. We use the following easy lemma:

Lemma 1. *Let $G = (V, E)$ be a connected undirected graph and let G' be a graph obtained from G by adding a new vertex $\tilde{v}$ and edges between $\tilde{v}$ and some vertices of G. Then $tw(G') \leq tw(G) + 1$.*

Proof. Given a tree decomposition Θ of G, we construct a tree decomposition Θ' for G' by adding to each bag of Θ the vertex $\tilde{v}$. It is easy to see that Θ' is indeed a tree decomposition for G' and that the width of Θ' is equal to the width of Θ plus one. Lemma 1 follows. □

Lemma 1 implies that the tree-width of the graph obtained from G by adding the cluster vertices and edges is at most the tree-width of G plus the number of cluster vertices, q. Since $q \leq \sqrt{2|\mathcal{N}|} + 1$ and $|\mathcal{N}|$ is fixed, Theorem 1 follows. □

Actually, this result was already proved in [11], but our proof is much simpler and much shorter. We also need to consider less clusterings than they do (since they have to enumerate all the clusterings containing at most $2|\mathcal{N}|$ clusters). Now, let us turn to the vertex variants.

In [1], it is shown that UVMinMC is polynomial-time solvable in trees and $\mathcal{NP}$-hard in unweighted graphs of bounded tree-width and bounded degree, and that VMinMC is $\mathcal{NP}$-hard in unweighted trees of bounded path-width and bounded degree. Moreover, recall that UVMinMC admits a PTAS for unweighted graphs of bounded tree-width. Therefore, one could wonder whether additional restrictions on the input graph could lead to polynomial cases more general than trees for UVMinMC.

In the remainder of this section, we show that this is very unlikely to happen, since the general scheme given in Section 2.2 can be used to show that UVMinMC is $\mathcal{NP}$-hard in unweighted triangular cacti of bounded path-width and bounded degree (even when each vertex appears only in a bounded number of bags of the path decomposition, i.e., even when the graph has *bounded persistence path-width* [6]), where a *triangular cactus* is a simple graph having no cycle of length 4 or more. The variable and clause

gadgets needed in the reduction are given in Fig. 1. It can easily be verified that they satisfy all the required properties.

The family of instances used in the proof is a family of unweighted triangular cacti of maximum degree 3, and, for this family, we can define a simple tree decomposition as follows: we put every variable gadget and every clause gadget in a bag (see Fig. 1). For example, $B1$ is the gadget of variable x_1 and $B8$ is the one of the clause $x_1 \vee \bar{x}_2 \vee x_3$. The tree of this decomposition is a path $B1, \ldots, Bj, \ldots$ (so, the path-width is bounded by $7-1=6$), where the Bi's are the bags, and each vertex appears at most 3 times. Hence:

Theorem 2. UVMinMC *is $\mathcal{NP}$-hard in unweighted triangular cacti of bounded persistence path-width and bounded degree.*

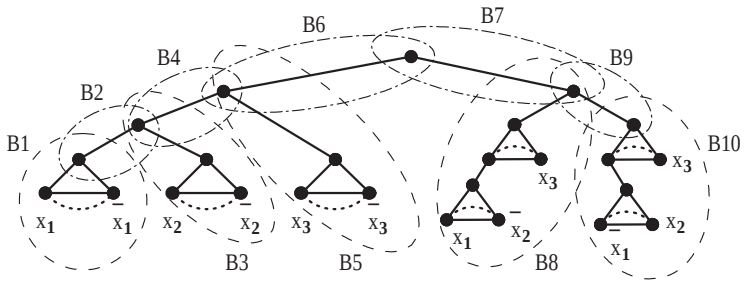

Figure 1: Reducing 3SAT to UVMinMC (the intra-gadget nets are in dotted lines) for the instance $(x_1 \vee \bar{x}_2 \vee x_3) \wedge (\bar{x}_1 \vee x_2 \vee x_3)$ ($Q = B2 \cup B4 \cup B6 \cup B7 \cup B9$)

4 Directed Graphs

In this section, we study the complexity of DEMinMC in directed acyclic graphs and in bounded tree-width and bounded degree unweighted digraphs, and, in particular, we investigate whether the ideas given in [1] for DEMinMC-SC could be applied to DEMinMC or not. It is worth noticing that no general polynomial reduction from a problem to the other is known; therefore none is theoretically "harder" than the other. However, we give such a reduction for the case where $|\mathcal{N}|$ is fixed:

Theorem 3. *When the number of terminals pairs $|\mathcal{N}|$ is fixed,* DEMinMC-SC *polynomially reduces to* DEMinMC.

Proof. The idea is that, given any source-sink pair (s_i, s_i'), either there remains no path from s_i to s_i' or there remains no path from s_i' to s_i in any optimal solution for DEMinMC-SC. Hence, one can try both possibilities for each net; this means that we need to solve $2^{|\mathcal{N}|}$ (i.e., $O(1)$ if $|\mathcal{N}|$ is fixed) instances of DEMinMC to solve the initial instance of DEMinMC-SC. □

One could also wonder whether the following transformation is a polynomial reduction from DEMinMC to DEMinMC-SC: given an instance I of DEMinMC, add an arc (s'_i, s_i) with a large weight for each i, in order to obtain an instance I' of DEMinMC-SC. Any optimal solution to I' is such that there remains no path from s_i to s'_i for each i (since the arcs (s'_i, s_i) are too heavy to belong to this solution), and thus is an optimal solution to $\hat{I}$, the instance of DEMinMC obtained from I by adding the arcs (s'_i, s_i). However, it should be noticed that $\hat{I}$ and I are not equivalent (because of the arcs (s'_i, s_i), there exist some paths in $\hat{I}$ that did not exist in I).

Some properties for DEMinMC are given in [13], where the authors conjecture that DEMinMC is not significantly simpler in directed acyclic graphs. Moreover, in [1], it is shown that DEMinMC-SC is $\mathcal{NP}$-hard in bounded tree-width and bounded degree unweighted digraphs. (Note that DEMinMC-SC is trivial in directed acyclic graphs, since the optimal value is always equal to 0.) Here, we first use the general scheme given in Section 2.2 to show that DEMinMC remains $\mathcal{NP}$-hard in bounded degree, unweighted, directed acyclic graphs (i.e., even when the tree-width is 0), while DEMinMTC is known to be tractable in directed acyclic graphs and DEMinMC is known to be tractable in directed trees [4].

To use this scheme, the main issue is to build directed gadgets having the required properties. The variable and clause gadgets that are needed in the proof are given in Fig. 2. It is easy to check that these directed acyclic gadgets satisfy the required properties (thanks to Remark 1). (Note that only the bold arcs are involved in "mutually exclusive ways", and so these arcs are the important ones.) This implies the following theorem:

Theorem 4. DEMinMC *is $\mathcal{NP}$-hard in directed acyclic graphs.*

Note that the directed acyclic graph used in our reduction is unweighted and has maximum degree 3. Moreover, the underlying undirected graph is a bipartite cactus, i.e., any edge belongs to at most one (even) cycle.

Also note that, if the maximum degree is 2, then the underlying undirected graph is the disjoint union of cycles and chains; therefore DEMinMC is tractable in this case (even if the directed graph is not acyclic) since it is tractable in paths and in rings [4].

Furthermore, if $|\mathcal{N}|$ is fixed and if the graph is unweighted and of maximum degree d (for fixed d), then DEMinMC is tractable. Indeed, a feasible solution can be obtained by selecting the arcs adjacent to the terminals (there are at most $d|\mathcal{N}|$ such arcs): hence, the value (and thus the size) of any optimal solution is bounded by $d|\mathcal{N}|$, and so we can compute an optimal solution by enumerating all the solutions with at most $d|\mathcal{N}|$ arcs and keeping the best feasible solution found.

Eventually, if the underlying undirected graph is bipartite (the bipartition being given by (V_1, V_2)), and if all the arcs are directed from V_1 to V_2 (i.e., if for any arc (u, v) of the graph, we have $u \in V_1$ and $v \in V_2$), then,

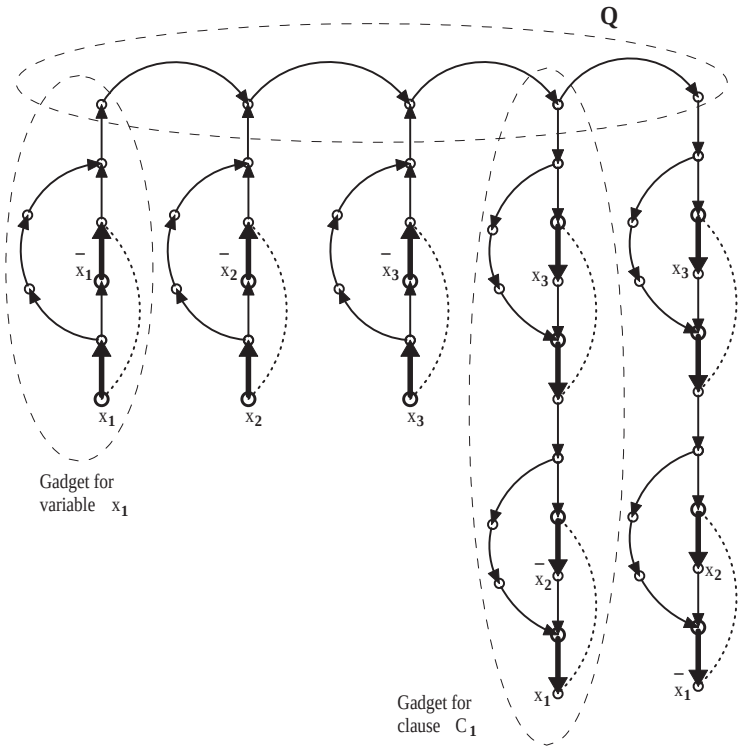

Figure 2: Reducing 3SAT to DEMINMC (the intra-gadget nets are in dotted lines) for the instance $C_1 \wedge C_2$, with $C_1 = x_1 \vee \bar{x}_2 \vee x_3$ and $C_2 = \bar{x}_1 \vee x_2 \vee x_3$

obviously, DEMINMC is tractable. Indeed, in this case, there is at most one arc linking s_i to s'_i for each i.

Actually, these graphs are a special case of the *layered digraphs* [13]: a *layered digraph* $D = (V, A)$ is a digraph whose vertex set V can be partitioned into $q \geq 2$ sets $V_1, \ldots, V_q$ such that, for each arc $a = (u, v) \in A$, we have $u \in V_i$ and $v \in V_{i+1}$ for some $i \in \{1, \ldots, q-1\}$. (Let the V_i's be the *layers* of D and let $p = \max_i |V_i|$.) Obviously, these graphs form a subclass of the directed acyclic graphs, and, for $q = 2$, we obtain the previous special case. Interestingly, solving DEMINMC in layered digraphs is in fact equivalent to solving it in directed acyclic graphs. Indeed, given a directed acyclic graph, we first compute a particular ordering on the vertices such that, for each arc (v_i, v_j), we have $i < j$ (this is called a *topological order*); then, we replace any arc (v_i, v_j) such that $j > i + 1$ by a path of length $j - i$ (all the arcs of this path having the same weight as the initial arc (v_i, v_j)).

Hence, Theorem 4 implies that DEMINMC is $\mathcal{NP}$-hard in layered digraphs. The proof of this theorem can even be easily adapted to show that DEMINMC remains $\mathcal{NP}$-hard in layered digraphs where either $q \geq 16$ is a

fixed integer (by contracting the path Q into a single vertex), or $p \geq 11$ is a fixed integer (this case generalizes the paths, where $p = 1$). (Note that the case where both p and q are fixed is trivial, since this implies that $|V|$ is fixed.) An interesting open problem would be to determine whether DEMINMC is tractable or not in layered digraphs, and hence in directed acyclic graphs, when $|\mathcal{N}|$ is fixed. We now study a more special case: we show that, when both p and $|\mathcal{N}|$ are fixed, DEMINMC can be solved in polynomial time by a simple dynamic programming algorithm.

The optimization function is the following: given $i \in \{1, \ldots, q\}$ and $|\mathcal{N}|$ sets $V_i^1 \subseteq V_i, \ldots, V_i^{|\mathcal{N}|} \subseteq V_i$, $f_i(V_i^1, \ldots, V_i^{|\mathcal{N}|})$ is equal to the value of a minimum weight set of arcs whose removal leaves (i) no path from s_j to s'_j for each j such that $s'_j \in V_{i'}$ for some $i' \leq i$, (ii) (at least) a path from s_j to any vertex in V_i^j for each j, and (iii) no path from s_j to any vertex in $V_i \setminus V_i^j$ for each j; this is equal to $+\infty$ if such a set of arcs does not exist. We initialize the computation of the f_i's by setting $f_1(V_1^1, \ldots, V_1^{|\mathcal{N}|})$ equal to 0 if $V_1^j = \{s_j\}$ (if $s_j \in V_1$) or $V_1^j = \emptyset$ (if $s_j \notin V_1$) for each j, and to $+\infty$ otherwise. Note that, for each i, the number of possible combinations $V_i^1, \ldots, V_i^{|\mathcal{N}|}$ (i.e., the number of values $f_i(\cdot)$) is $O(2^{p|\mathcal{N}|})$. Our algorithm proceeds by computing the values $f_{i+1}(\cdot)$ from $f_i(\cdot)$ for each i, and returns $\min_{(V_q^1, \ldots, V_q^{|\mathcal{N}|})} f_q(V_q^1, \ldots, V_q^{|\mathcal{N}|})$.

For each $i < q$, let $A_{i,i+1}$ be the set of arcs between V_i and V_{i+1}, and, given $B \subseteq A_{i,i+1}$, let $w(\bar{B})$ be the sum of the weights of the arcs in $A_{i,i+1} \setminus B$. Moreover, given two integers $i > 0$ and $j > 0$ and two sets $B \subseteq A_{i,i+1}$ and $V_i^j \subseteq V_i$, we define $V_i^j \triangleright B = \{s_j\}$ if $s_j \in V_{i+1}$ and $V_i^j \triangleright B = \{v \in V_{i+1}$ such that there is an arc $(u, v) \in B$ with $u \in V_i^j\}$ otherwise. (Note that, if $s_j \notin V_{i+1}$, then $V_i^j \triangleright B = \emptyset$ when either $V_i^j = \emptyset$ or $B = \emptyset$.)

The relationship between the $f_{i+1}(\cdot)$'s and the $f_i(\cdot)$'s can then be described as follows. Given $(V_{i+1}^1, \ldots, V_{i+1}^{|\mathcal{N}|})$ and an integer $i > 0$, if there exist $(V_i^1, \ldots, V_i^{|\mathcal{N}|}, B)$ such that $\forall j$, $V_i^j \triangleright B = V_{i+1}^j$, and if, in addition, $s'_j \notin V_{i+1}^j$ for each j, then we have:

$$f_{i+1}(V_{i+1}^1, \ldots, V_{i+1}^{|\mathcal{N}|}) = \min_{(V_i^1, \ldots, V_i^{|\mathcal{N}|}, B) \ | \ \forall j, \ V_i^j \triangleright B = V_{i+1}^j} \Big(f_i(V_i^1, \ldots, V_i^{|\mathcal{N}|}) + w(\bar{B}) \Big)$$

Otherwise, we have $f_{i+1}(V_{i+1}^1, \ldots, V_{i+1}^{|\mathcal{N}|}) = +\infty$. The number of combinations $(V_i^1, \ldots, V_i^{|\mathcal{N}|}, B)$ is at most $O(2^{p|\mathcal{N}|} 2^{p^2})$ (since $|A_{i,i+1}| \leq |V_i||V_{i+1}| \leq p^2$), and we keep only $O(2^{p|\mathcal{N}|})$ of them (since there are at most $O(2^{p|\mathcal{N}|})$ distinct vectors $(V_i^1 \triangleright B, \ldots, V_i^{|\mathcal{N}|} \triangleright B)$), so the f_i's can be computed in $O(2^{p(|\mathcal{N}|+p)})$ time for each i. Thus, our algorithm is FPT [7] with respect to the pair of parameters $(p, |\mathcal{N}|)$, because the overall complexity is $O(2^{p(|\mathcal{N}|+p)} q)$. Actually, when p and $|\mathcal{N}|$ are fixed, its running time is even linear in q (i.e., in the size of the graph).

Now, what about the PTAS for DEMinMC-SC in bounded tree-width and bounded degree unweighted digraphs? Unfortunately, it cannot be adapted for DEMinMC, since, when proving that the dynamic programming algorithm does provide a feasible solution (i.e., that it outputs a valid multicut), a key argument (see [1, p. 348]) uses the fact that any directed cycle starts and ends in the same set W_r, since it starts and ends at the same vertex. Clearly, such an argument cannot be applied to our variant, since we deal with paths (the two endpoints of a path can belong to any sets W_r and W_s) instead of directed cycles.

Actually, we show that DEMinMC is $\mathcal{APX}$-hard in bounded tree-width and bounded degree unweighted digraphs (while the undirected case admits a PTAS), implying that DEMinMC-SC is easier than DEMinMC even in this special case (since there exists a PTAS). To do this, we present an approximation-preserving reduction from Vertex Cover in bounded-degree graphs, which is known to be $\mathcal{APX}$-hard [16]. Our result matches the best inapproximability result known for DEMinMC in *unrestricted digraphs* that is based on the assumption that $\mathcal{P} \neq \mathcal{NP}$ (the $\Omega\left(\frac{\log n}{\log \log n}\right)$ inapproximability bound of Chuzhoy and Khanna [3] being based on a stronger complexity assumption, and the $\Omega(1)$ inapproximability bound of Chawla et al. [2] being based on a different one, the *Unique Games Conjecture*).

Assume we are given an undirected graph G with n vertices and bounded maximum degree. A vertex cover of G is a set of vertices C of G such that for each edge $e = (u, v)$ of G, either $u \in C$ or $v \in C$. We orient G as follows: we arbitrarily number the n vertices $v_1, \ldots, v_n$, and then we transform each edge (v_i, v_j) into an arc (v_i, v_j) (if $i < j$) or an arc (v_j, v_i) (if $j < i$). This way, $v_1, v_2, \ldots, v_n$ defines a topological ordering of the vertices, and the directed graph D we obtain is acyclic (and, therefore, has tree-width 0). Moreover, the vertex covers in G are the (unrestricted) vertex multiterminal cuts in D (any vertex being a terminal), and thus the transformation applied so far proves that DUVMinMTC is $\mathcal{APX}$-hard in directed acyclic graphs (while DEMinMTC is tractable in this case [4]). Now, let us use D to obtain a DEMinMC instance. We add n new vertices $t_1, t_2, \ldots, t_n$, and, for each i, we link v_i and t_i by a gadget with five arcs (see Fig. 3: for an i, we add two new vertices a_i and b_i, and the five arcs are (v_i, a_i), (b_i, v_i), (a_i, b_i), (t_i, a_i) and (b_i, t_i)). This way, we obtain a new graph H. Then, it is easy to see that we have a vertex multiterminal cut of size S in D (and thus a vertex cover of size S in G) if and only if we have an arc multicut of size S in H, where the nets are (t_i, t_j) for each arc (v_i, v_j) of D. Indeed, given a vertex cover C of size S for G, we can construct an arc multicut of size S in H as follows: for each i, pick (a_i, b_i) in the cut if and only if $v_i \in C$. Conversely, given an arc multicut of size S for H, either only arcs of the form (a_i, b_i) are in the cut, or we can transform this cut into a new cut of size at most S

containing only such arcs: if an arc (v_i, v_j) is cut, then replace it by (a_i, b_i); if an arc of the gadget linking v_i and t_i is cut, then also replace it by (a_i, b_i). We can now define a vertex cover of size S in G by including all the v_i's such that (a_i, b_i) is in the arc multicut for H. This completes the description of the approximation-preserving reduction.

It is easily seen that H is unweighted and has bounded in- and out-degrees (since G has bounded degree). It remains to show that H has bounded tree-width. We construct an arboreal decomposition Θ for H as follows. The rooted tree of the decomposition is a directed path with n vertices. The set W_r associated with the r^{th} vertex of this path is $W_r = \{v_r, a_r, b_r, t_r\}$ (W_1 being the root). Moreover, the sets X_e associated with the arcs of this path are empty. Then, the W_r's do define a partition of the vertices of H. Assume all the vertices $v_1, v_2, \ldots, v_n$ of D lie on a horizontal line in this order: then, by the way we constructed D, any arc between two v_i's will be oriented from left to right. Hence, it is easy to see that, for any r, any path that goes from a vertex in W_{r_1} with $r_1 \geq r$ to a vertex in W_{r_2} with $r_2 \geq r_1$ does not use any vertex contained in a set lying on the left of r (i.e., in a set $W_{r'}$ with $r' < r$). Thus, Θ is indeed an arboreal decomposition of H. Moreover, its width is $4 - 1 = 3$. This implies:

Theorem 5. DEMINMC *is $\mathcal{APX}$-hard in bounded tree-width and bounded degree unweighted digraphs.*

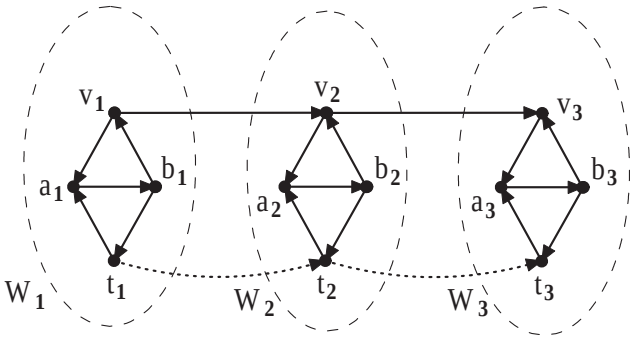

Figure 3: Transforming a graph G (here, a path with 3 vertices v_1, v_2, v_3) into H (the nets are in dotted lines), to reduce VERTEX COVER to DEMINMC

To conclude, we provide a table summarizing the main complexity (and inapproximability) results known for MINMC (and all its variants) in graphs and digraphs of bounded tree-width.

Problem	Bounded tree-width	Bounded degree	Unweighted	Fixed $\lvert\mathcal{N}\rvert$	Result
EMinMC	✓		✓		$\mathcal{APX}$-hard in stars [10]
MinMC			✓	✓	$\mathcal{APX}$-hard [5]
EMinMTC	✓				Polynomial [5]
EMinMC	✓			✓	***Polynomial***
UVMinMC	✓		✓		PTAS [1]
UVMinMC	✓	✓	✓		$\mathcal{NP}$-hard in series-parallel graphs [1]
UVMinMC	✓		✓		Polynomial in trees [1]
UVMinMC	✓	✓	✓		***$\mathcal{NP}$-hard even in cacti of bounded path-width***
DEMinMC	✓				Polynomial in ditrees [4]
DEMinMTC	✓				Polynomial in DAG [4]
DEMinMC	✓	✓	✓		***$\mathcal{NP}$-hard in DAG (directed cacti)***
DEMinMC	✓			✓	***Polynomial in layered digraphs if fixed-sized layers***
EMinMC	✓	✓	✓		PTAS & $\mathcal{NP}$-hard ($\mathcal{APX}$-hard if weighted or arbitrary degree or tree-width) [1]
DEMinMC-SC	✓	✓	✓		As above
(D)UVMinMTC	✓	✓	✓		***$\mathcal{APX}$-hard even in DAG***
DEMinMC	✓	✓	✓		***$\mathcal{APX}$-hard***

Table 1: Summary of the various complexity results known for MinMC (the results proved in the present paper are indicated in bold letters)

References

[1] Călinescu, G., Fernandes, C.G., Reed, B.: Multicuts in unweighted graphs and digraphs with bounded degree and bounded tree-width. Journal of Algorithms **48** (2003) 333–359.

[2] Chawla, S., Krauthgamer, R., Kumar, R., Rabani, Y., Sivakumar, D.: On the hardness of approximating multicut and sparsest-cut. In: Proceedings of the Computational Complexity Conference (2005).

[3] Chuzhoy, J., Khanna, S.: Hardness of Cut Problems in Directed Graphs. In: STOC'06 (2006) 527–536.

[4] Costa, M.-C., Létocart, L., Roupin, F.: Minimal multicut and maximal integer multiflow: a survey. European Journal of Operational Research **162** (2005) 55–69.

[5] Dahlhaus, E., Johnson, D.S., Papadimitriou, C.H., Seymour, P.D., Yannakakis, M.: The complexity of multiterminal cuts. SIAM Journal on Computing **23** (1994) 864–894.

[6] Downey, R.G., McCartin, C.: Bounded Persistence Pathwidth. In: Atkinson, M., Dehne, F. (eds.): Eleventh Computing: The Australasian Theory Symposium (CATS'2005), Newcastle, Australia. CRPIT **41** (2005) 51–56.

[7] Downey, R.G., Fellows, M.R.: Parameterized Complexity. Springer-Verlag, New York (1999).

[8] Garey, M.R., Johnson, D.S.: Computers and Intractability: a Guide to the Theory of NP-completeness. W.H. Freeman and Co., San Fransisco (1979).

[9] Garg, N., Vazirani, V.V., Yannakakis, M.: Approximate max-flow min-(multi)cut theorems and their applications. SIAM Journal on Computing **25** (1996) 235–251.

[10] Garg, N., Vazirani, V.V., Yannakakis, M.: Primal-dual approximation algorithms for integral flow and multicut in trees. Algorithmica **18** (1997) 3–20.

[11] Guo, J., Hüffner, F., Kenar, E., Niedermeier, R., Uhlmann, J.: Complexity and Exact Algorithms for Multicut. In: SOFSEM'06, Lecture Notes in Computer Science **3831** (2006) 303–312.

[12] Gupta, A.: Improved results for directed multicut. In: 14th ACM-SIAM Symposium on Discrete Algorithms (2003) 454–455.

[13] Hajiaghayi, M., Leighton, T.: On the max-flow min-cut ratio for directed multicommodity flows. MIT Technical Report 910 (2003), 5p.

[14] Johnson, T., Robertson, N., Seymour, P., Thomas, R.: Directed Tree-width. Journal of Combinatorial Theory Series B, **82** (2001) 138–154.

[15] Klein, P., Plotkin, S., Rao, S., Tardos, É.: Approximation Algorithms for Steiner and Directed Multicuts. Journal of Algorithms **22** (1997) 241–269.

[16] Papadimitriou, C., Yannakakis, M.: Optimization, approximation, and complexity classes. J. Comput. and System Sciences **43** (1991) 425–440.

[17] Robertson, N., Seymour, P.D.: Graph minors II: Algorithmic aspects of tree-width. Journal of Algorithms **7** (1986) 309–322.