Efficient Enumerations for Minimal Multicuts and Multiway Cuts

Kazuhiro Kurita
National Institute of Informatics, Tokyo, Japan
kurita@nii.ac.jp

Yasuaki Kobayashi
Kyoto University, Japan
kobayashi@iip.ist.i.kyoto-u.ac.jp

Abstract

Let $G = (V, E)$ be an undirected graph and let $B \subseteq V \times V$ be a set of terminal pairs. A node/edge multicut is a subset of vertices/edges of G whose removal destroys all the paths between every terminal pair in B. The problem of computing a *minimum* node/edge multicut is NP-hard and extensively studied from several viewpoints. In this paper, we study the problem of enumerating all *minimal* node multicuts. We give an incremental polynomial delay enumeration algorithm for minimal node multicuts, which extends an enumeration algorithm due to Khachiyan et al. (Algorithmica, 2008) for minimal edge multicuts.

Important special cases of node/edge multicuts are node/edge *multiway cuts*, where the set of terminal pairs contains every pair of vertices in some subset $T \subseteq V$, that is, $B = T \times T$. We improve the running time bound for this special case: We devise a polynomial delay and exponential space enumeration algorithm for minimal node multiway cuts and a polynomial delay and space enumeration algorithm for minimal edge multiway cuts.

2012 ACM Subject Classification Mathematics of computing → Graph algorithms

Keywords and phrases Multicuts, Multiway cuts, Enumeration algorithms

Digital Object Identifier 10.4230/LIPIcs.MFCS.2020.60

Related Version `https://arxiv.org/abs/2006.16222`

Funding *Kazuhiro Kurita*: JSPS KAKENHI Grant Number JP19J10761.
Yasuaki Kobayashi: JSPS KAKENHI Grant Numbers JP20K19742 and JP20H00595.

1 Introduction

Let $G = (V, E)$ be an undirected graph and let B be a set of pairs of vertices of V. We call a pair in B a *terminal pair* and the set of vertices in B is denoted by $T(B)$. A *node multicut* of (G, B) is a set of vertices $M \subseteq V \setminus T(B)$ such that there is no path between any terminal pair of B in the graph obtained by removing the vertices in M. A *edge multicut* of (G, B) is defined as well: the set of edges whose removal destroys all the paths between every terminal pair. The minimum node/edge multicut problem is of finding a smallest cardinality node/edge multicut of (G, B). When $B = T \times T$ for some $T \subseteq V$, the problems are called the minimum node/edge multiway cut problems, and a multicut of (G, B) is called a *multiway cut* of (G, T).

These problems are natural extensions of the classical minimum s-t separator/cut problems, which can be solved in polynomial time using the augmenting path algorithm. Unfortunately, these problems are NP-hard [11] even for planar graphs and for general graphs with fixed $|T| \geq 3$. Due to numerous applications (e.g. [14,23,36]), a lot of efforts have been devoted to solving these problems from several perspectives such as approximation algorithms [1,9,17,18,24], parameterized algorithms [10,20,30,32,40], and restricting input [3,6,11,21,27,31].

In this paper, we tackle these problems from yet another viewpoint. Our focus is *enumeration.* Since the problems of finding a *minimum* node/edge multicut/multiway cut are all intractable, we rather enumerate *minimal* edge/node multicuts/multiway cuts instead. We say that a node/edge multicut M of (G, B) is minimal if M' is not a node/edge multiway cut of (G, B) for every proper subset $M' \subset M$, respectively. Minimal node/edge multiway cuts are defined accordingly. Although finding a minimal node/edge multicut is easy, our goal is to enumerate *all* the minimal edge/node multicuts/multiway cuts of a given graph G and terminal pairs B. In this context, there are several results related to our problems.

There are linear delay algorithms for enumerating all minimal s-t (edge) cuts [33, 38], which is indeed a special case of our problems, where T contains exactly two vertices s and t. Here, an enumeration algorithm has *delay* complexity $f(n)$ if the algorithm outputs all the solutions without duplication and for each pair of consecutive two outputs (including preprocessing and postprocessing), the running time between them is upper bounded by $f(n)$, where n is the size of the input. For the node case, the problem of enumerating all minimal s-t (node) separators has received a lot of attention and numerous efforts have been done for developing efficient algorithms [28, 35, 37] due to many applications in several fields [4, 13, 15]. The best known enumeration algorithm for minimal s-t separators was given by Tanaka [37], which runs in $O(nm)$ delay and $O(n)$ space, where n and m are the number of vertices and edges of an input graph, respectively.

Khachiyan et al. [25] studied the minimal edge multicut enumeration problem. They gave an efficient algorithm for this problem, which runs in *incremental polynomial time* [22], that is, if $\mathcal{M}$ is a set of minimal edge multicuts of (G, B) that are generated so far, then the algorithm decides whether there is a minimal edge multicut of G not included in $\mathcal{M}$ in time polynomial in $|V|+|E|+|\mathcal{M}|$. Moreover, if such a minimal edge multicut exists, the algorithm outputs one of them within the same running time bound. As we will discuss in the next section, this problem is a special case of the node counterpart and indeed a generalization of the minimal edge multiway cut enumeration problem. Therefore, this algorithm also works for enumerating all minimal edge multiway cuts. However, there can be exponentially many minimal edge multicuts in a graph. Hence, the delay of their algorithm cannot be upper bounded by a polynomial in terms of input size. To the best of our knowledge, there is no known non-trivial enumeration algorithm for minimal node multiway cuts.

Let $(G = (V, E), B)$ be an instance of our enumeration problems. In this paper, we give polynomial delay or incremental polynomial delay algorithms.

▶ **Theorem 1.** *There is an algorithm which enumerates all the minimal node and edge multiway cuts of* (G, B) *in* $O(|T(B)| \cdot |V| \cdot |E|)$ *and* $O\big(|T(B)| \cdot |V| \cdot |E|^2\big)$ *delay, respectively.*

The algorithm in Theorem 1 requires exponential space to avoid redundant outputs. For the edge case, we can simultaneously improve the time and space consumption.

▶ **Theorem 2.** *There is an algorithm which enumerates all the minimal edge multiway cuts of* (G, B) *in* $O(|T(B)| \cdot |V| \cdot |E|)$ *delay in polynomial space.*

For the most general problem among them (i.e., the minimal node multicut enumeration problem), we give an incremental polynomial time algorithm.

▶ **Theorem 3.** *There is an algorithm of finding, given a set of minimal node multicuts* $\mathcal{M}$ *of* (G, B)*, a minimal node multicut* M *of* (G, B) *with* $M \notin \mathcal{M}$ *if it exists and runs in time* $O(|\mathcal{M}| \cdot poly(n))$.

The first and second results simultaneously improve the previous incremental polynomial running time bound obtained by applying the algorithm of Khachiyan et al. [25] to the edge multiway cut enumeration and extends enumeration algorithms for minimal s-t cuts [33, 38] and minimal a-b separators [37][1]. The third result extends the algorithm of Khachiyan et al. to the node case. Since enumerating minimal node multicuts is at least as hard as enumerating minimal node multiway cuts and enumerating minimal node multiway cuts is at least as hard as enumerating minimal edge multiway cuts (See Proposition 4 for details), this hierarchy directly reflects on the running time of our algorithms.

The basic idea behind these results is that we rather enumerate a particular collection of partitions/disjoint subsets of V than directly enumerating minimal edge/node multicuts/multiway cuts of (G, B). It is known that an s-t edge cut of G is minimal if and only if the bipartition (V_1, V_2) naturally defined from the s-t cut induces connected subgraphs of G, that is, $G[V_1]$ and $G[V_2]$ are connected [12]. For minimal a-b separators, a similar characterization is known using full components (see [19], for example). These facts are highly exploited in enumerating minimal s-t cuts [33, 38] or minimal a-b separators [37], and can be extended for our cases (See Sections 3, 4, and 5). To enumerate such a collection of partitions/disjoint subsets of V in the claimed running time, we use three representative techniques: the *proximity search paradigm* due to Conte and Uno [8] for the exponential space enumeration of minimal node multiway cuts, the *reverse search paradigm* due to Avis and Fukuda [2] for polynomial space enumeration of minimal edge multiway cuts, and the *supergraph approach*, which appeared implicitly and explicitly in the literature [7, 8, 25, 34], for the incremental polynomial time enumeration of minimal node or edge multicuts. These approaches basically define a (directed) graph on the set of solutions we want to enumerate. If we appropriately define some adjacency relation among the vertices (i.e. the set of solutions) so that the graph is (strongly) connected, then we can enumerate all solutions from a specific or arbitrary solution without any duplication by traversing this (directed) graph. The key to designing the algorithms in Theorem 1 and 2 is to ensure that every vertex in the graphs defined on the solutions has a polynomial number of neighbors.

We also consider a generalization of the minimal node multicut enumeration, which we call the minimal Steiner node multicut enumeration. We show that this problem is at least as hard as the minimal transversal enumeration on hypergraphs.

Due to the space limitation, proofs (marked ★) are omitted and can be found in the full version [29].

2 Preliminaries

In this paper, we assume that a graph $G = (V, E)$ is connected and has no self-loops and no parallel edges. Let $X \subseteq V$. We denote by $G[X]$ the subgraph of G induced by X. The neighbor set of X is denoted by $N_G(X)$ (i.e. $N_G(X) = \{y \in V \setminus X : x \in X \land \{x, y\} \in E\}$ and the closed neighbor set of X is denoted by $N_G[X] = N(X) \cup X$. When X consists of a single vertex v, we simply write $N_G(v)$ and $N_G[v]$ instead of $N_G(\{v\})$ and $N_G[\{v\}]$, respectively. If there is no risk of confusion, we may drop the subscript G. For a set of vertices $U \subseteq V$ (resp. edges $F \subseteq E$), the graph obtained from G by deleting U (resp. F) is denoted by $G - U$ (resp. $G - F$).

Let B be a set of pairs of vertices in V. We denote by $T(B) = \{s, t : \{s, t\} \in B\}$. A vertex in $T(B)$ is called a *terminal*, a pair in B is called a set of *terminal pairs*, and $T(B)$ is called a *terminal set* or *terminals*. When no confusion can arise, we may simply use T to denote the terminal set. A set of edges $M \subseteq E$ is an *edge multicut* of (G, B) if no pair of terminals in B is connected in $G - M$. When B is clear from the context, we simply call

[1] However, our algorithm requires exponential space for minimal node multiway cuts, whereas Takata's algorithm [37] runs in polynomial space.

M an edge multicut of G. An edge multicut M is *minimal* if every proper subset $M' \subset M$ is not an edge multicut of G. Note that this condition is equivalent to that $M \setminus \{e\}$ is not an edge multicut of G for any $e \in M$. Analogously, a set of vertices $X \subseteq V \setminus T$ is a *node multicut* of G if there is no paths between any terminal pair of B in $G - X$. The minimality for node multicuts is defined accordingly.

The *demand graph for B* is a graph defined on $T(B)$ in which two vertices s and t are adjacent to each other if and only if $\{s,t\} \in B$. When B contains a terminal pair $\{s,t\}$ for any distinct $s,t \in T(B)$, that is, the demand graph for B is a complete graph, a node/edge multicut is called a *node/edge multiway cut* of G.

Let $G = (V, E)$ be a graph and let B be a set of terminal pairs. Let G' be the graph obtained from the line graph of G by adding a terminal t' for each $t \in T$ and making t' adjacent to each vertex corresponding to an edge incident to t in G.

▶ **Proposition 4.** *Let $M \subseteq E$. Then, M is an edge multicut of G if and only if M is a node multicut of G'.*

By Proposition 4, designing an enumeration algorithm for minimal node multicuts/multiway cuts, it allows us to enumerate minimal edge multicuts/multiway cuts as well. However, the converse does not hold in general.

3 Incremental polynomial time enumeration of minimal node multicuts

In this section, we design an incremental polynomial time enumeration algorithm for minimal node multicuts. Let $G = (V, E)$ and let B be a set of terminal pairs.

For a (not necessarily minimal) node multicut M of G, there are connected components $C_1, C_2, \ldots, C_\ell$ in $G - M$ such that each component contains at least one terminal but no component has a terminal pair in B. Note that there can be components of $G - M$ not included in $\{C_1, \cdots, C_\ell\}$. The following lemma characterizes the minimality of node multicut in this way.

▶ **Lemma 5** (★). *A set of vertices $M \subseteq V \setminus T$ of G is a minimal node multicut if and only if there are ℓ connected components $C_1, C_2, \ldots, C_\ell$ in $G - M$, each of which includes at least one terminal of T, such that (1) there is no component which includes both vertices in a terminal pair and (2) for any $v \in M$, there is a terminal pair (s_i, t_i) such that both components including s_i and t_i have a neighbor of v.*

From a minimal node multicut M of G, we can uniquely determine the set $\mathcal{C}$ of ℓ components satisfying the conditions in Lemma 5, and vice-versa. Given this, we denote by $\mathcal{C}_M$ the set of components corresponding to a minimal multicut M. From now on, we may interchangeably use $M \subseteq V \setminus T$ and $\mathcal{C}_M$ as a minimal node multicut of G. For a (not necessarily minimal) node multicut M of G, we also use $\mathcal{C}_M$ to denote the set of connected components $\{C_1, \ldots, C_\ell\}$ of $G - M$ such that each component contains at least one terminal but no component has a terminal pair.

We enumerate all the minimal node multicuts of G using the supergraph approach [7, 8, 25, 26]. To this end, we define a directed graph on the set of all the minimal node multicuts of G, which we call a *solution graph*. The outline of the supergraph approach is described in Algorithm 1. The following "distance" function plays a vital role for our enumeration algorithm: For (not necessarily minimal) node multicuts M and M' of G,

$$\mathtt{dist}(M, M') = \sum_{C' \in \mathcal{C}_{M'}} |C' \setminus \mathtt{mcc}(C', M)|,$$

Algorithm 1 Traversing a solution graph $\mathcal{G}$ using a breadth-first search.

```
Procedure Traversal($\mathcal{G}$)
    $S \leftarrow$ an arbitrary solution
    $\mathcal{Q},\mathcal{U} \leftarrow \{S\}, \emptyset$
    while $\mathcal{Q} \neq \emptyset$ do
        Let $S$ be a solution in $\mathcal{Q}$
        Output $S$        //We do not output here for minimal node multicuts
        Delete $S$ from $\mathcal{Q}$
        for $S' \in$ Neighborhood($S$,$\mathcal{U}$) do
            if $S' \notin \mathcal{U}$ then $\mathcal{Q},\mathcal{U} \leftarrow \mathcal{Q} \cup \{S'\}, \mathcal{U} \cup \{S'\}$
```

where $\texttt{mcc}\,(C', M)$ is the component C of $G - M$ minimizing $|C' \setminus C|$. If there are two or more components C minimizing $|C' \setminus C|$, we define $\texttt{mcc}\,(C', M)$ as the one having a smallest vertex with respect to some prescribed order on V among those components. It should be mentioned that the function $\texttt{dist}$ is not the actual distance in the solution graph which we will define later. Note moreover that this value can be defined between two non-minimal node multicuts as $\mathcal{C}_M$ is well-defined for every node multicut M of G. Let M, M', and M'' be (not necessarily minimal) node multicuts of G. Then, we say that M is *closer than M' to M''* if $\texttt{dist}(M, M'') < \texttt{dist}(M', M'')$.

▶ **Lemma 6** (★). *Let M and M' be minimal node multicuts of G. Then, M is equal to M' if and only if $\texttt{dist}(M, M') = 0$.*

From a node multicut M of G, a function μ maps M to an arbitrary minimal node multicut $\mu(M) \subseteq M$. In this paper, we define $\mu(M)$ as follows: If there is a vertex v such that $M \setminus \{v\}$ is a node multicut, we remove v. When there are two or more such vertices, we pick a vertex with the minimum index. Clearly, this function computes a minimal node multicut of G in polynomial time.

▶ **Lemma 7** (★). *Let M be a node multicut of G and M' a minimal node multicut of G. Then, $\texttt{dist}(\mu(M), M') \leq \texttt{dist}(M, M')$ holds.*

To complete the description of Algorithm 1, we need to define the neighborhood of each minimal node multicut of G. To enumerate all the minimal node multicut of G, we want to ensure that the solution graph is strongly connected. To do this, we exploit $\texttt{dist}$ as follows. Let M and M' be distinct minimal node multicuts of G. We will define the neighborhood of M in such a way that it should contain at least one minimal node multiway cut M'' of G that is closer than M to M'. This allows to eventually have M' from M with Algorithm 1. The main difficulty is that the neighborhood of M contains such M'' for every M', which will be described in the rest of this section.

To make the discussion simpler, we use the following two propositions. Here, for an edge e of G, we let G/e denote the graph obtained from G by contracting edge e. We use v_e to denote the newly introduced vertex in G/e.

▶ **Proposition 8.** *Let t_1 be a terminal adjacent to another terminal t_2 in G. Suppose $\{t_1, t_2\}$ is not included in B. Then, M is a minimal node multicut of (G, B) if and only if it is a minimal node multicut of $(G/e, B')$, where $e = \{t_1, t_2\}$ and B' is obtained by replacing t_1 and t_2 in B with the new vertex v_e in G/e.*

If G has an adjacent terminal pair in B, then obviously there is no node multicut of G. By Proposition 8, G has no adjacent terminals.

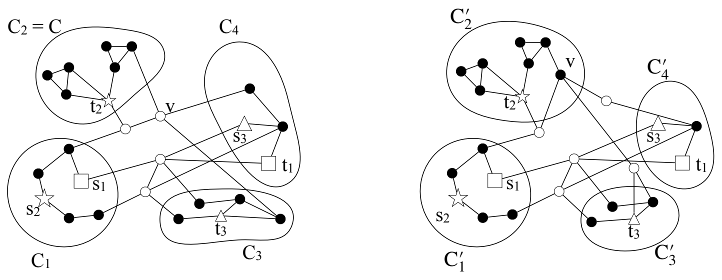

Figure 1 This figure illustrates an example of Lemma 10. White circles represent vertices in node multicuts and pairs of stars, squares, and triangles represent terminal pairs. The left and right pictures depict a minimal node multicut M and a node multicut M''.

▶ **Proposition 9.** *If there is a vertex v of G such that $N(v)$ contains a terminal pair $\{s, t\}$, then for every node multicut M of G, we have $v \in M$. Moreover, M is a minimal node multicut of (G, B) if and only if $M \setminus \{v\}$ is a minimal node multicut of $(G - \{v\}, B)$.*

From the above two propositions, we assume that there is no pair of adjacent terminals and no vertex including a terminal pair in B as its neighborhood. To define the neighborhood of M, we distinguish two cases.

▶ **Lemma 10 (★).** *Let M and M' be distinct minimal node multicuts of G, let $C' \in \mathcal{C}_{M'}$, and let $C = \texttt{mcc}(C', M)$. Suppose there is a vertex $v \in N(C) \cap C' \subseteq M$ such that $N[v] \cup C$ has no terminal pair in B. Let $T_v = N(v) \cap T$. Then $M'' = (M \setminus \{v\}) \cup (N(T_v \cup \{v\}) \setminus C)$ is a node multicut of G. Moreover, $\mu(M'')$ is closer than M to M'.*

Figure 1 illustrates an example of M and M'' in Lemma 10.

If there is a terminal pair $\{s, t\} \in B$ in $G[C \cup N[v]]$, M'' defined in Lemma 10 is not a node multicut of G since s and t are contained in the connected component $C \cup T_v \cup \{v\}$ of $G - M''$ (see Figure 2). In this case, we have to separate all terminal pairs in this component.

▶ **Lemma 11 (★).** *Let M and M' be two distinct minimal node multicuts of G, let $C' \in \mathcal{C}_{M'}$, and let $C = \texttt{mcc}(C', M)$. Suppose there is a vertex $v \in N(C) \cap C' \subseteq M$ such that $N[v] \cup C$ contains some terminal pair in B. Let $T_v = N(v) \cap T$. Then, $M'' = (M \setminus \{v\}) \cup (N(T_v \cup \{v\}) \setminus C) \cup (C \cap M')$ is a node multicut and $\mu(M'')$ is closer than M to M'.*

Now, we formally define the neighborhood of a minimal node multicut M in the solution graph. Our goal is to ensure the strong connectivty of the solution graph. For each component C and $v \in N(C)$, the neighborhood of M contains $\mu((M \setminus \{v\}) \cup (N(T_v \cup \{v\}) \setminus C))$ if $N[v] \cup C$ has no terminal pair in B and $\mu((M \setminus \{v\}) \cup (N(T_v \cup \{v\}) \setminus C) \cup (C \cap M'))$ otherwise. By Lemmas 10 and 11, this neighborhood relation ensures that the solution graph is strongly connected, which allows us to enumerate all the minimal node multicuts of G from an arbitrary one using Algorithm 1. However, there is an obstacle: We have to generate the neighborhood without knowing M' for the case where $N[v] \cup C$ has a terminal pair. To this end, we show that computing the neighborhood for this case can be reduced to enumerating the minimal a-b separators of a graph.

Suppose that $N[v] \cup C$ has a terminal pair. Let M' be an arbitrary node multicut of G. An important observation is that $C \cap M'$ is a node multicut of $(G[C \cup T_v \cup \{v\}], \{\{s, t\} : \{s, t\} \in B, s \in T_v, t \in C\})$. Since C is a component of $G - M$, by Proposition 9, one of the terminals in each terminal pair contained in $N[v] \cup C$ belongs to $N(v) \cap T$ and the other one belongs to C. Thus, every path between those terminal pairs pass through v in $G[C \cup T_v \cup \{v\}]$

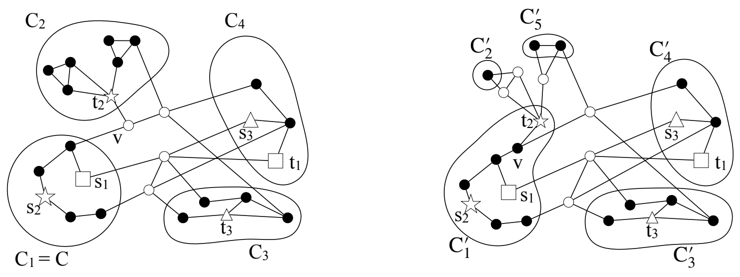

Figure 2 This figure illustrates an example of Lemma 11. $(M \setminus \{v\}) \cup (N(T_v \cup \{v\}) \setminus C_1))$ is not a node multicut of G, and then we additionally have to separate a pair of terminals (represented by stars) in the component $C'_1 = C_1 \cup T_v \cup \{v\}$.

and $v \notin M'$. It implies that $C \cap M'$ is a node multicut of $(G[C \cup \{v\}], \{\{v,t\} : \{s,t\} \in B, t \in C\})$. Let $H = G[C \cup \{v\}]$ and let $B' = \{\{v,t\} : \{s,t\} \in B, t \in C\})$. Moreover, if we have two distinct minimal node multicuts M_1 and M_2 of (H, B'), minimal node multicuts $\mu((M \setminus \{v\}) \cup (N(T_v \cup \{v\}) \setminus C) \cup (C \cap M_i))$, for $i = 1, 2$, are distinct since function μ does not remove any vertex in M_1 and M_2.

Now, our strategy is to enumerate minimal node multicuts $\mu(C \cap M')$ of (H, B') for all M'. This subproblem is not easier than the original problem at first glance. However, this instance (H, B') has a special property that the demand graph for B' forms a star. From this property, we show that this problem can be reduced to the minimal a-b separator enumeration problem.

▶ **Lemma 12** (★). *Let $H = G[C \cup \{v\}]$ and let $B' = \{\{v,t\} : \{s,t\} \in B, t \in C\}$. Let H' be the graph obtained from H by identifying all the vertices of $T(B') \setminus \{v\}$ into a single vertex v_t. Then, $M \subseteq (C \cup \{v\}) \setminus T(B')$ is minimal node multicut of (H, B') if and only if M is a minimal v-v_t separator of H'.*

By Lemma 12, we can enumerate $\mu(C \cap M')$ for every minimal node multicut M' of G by using the minimal a-b separator enumeration algorithm of Takata [37]. Moreover, as observed above, for any distinct minimal v-v_t separators S_1 and S_2 in H', we can generate distinct minimal node multicuts $\mu((M \setminus \{v\}) \cup (N(T_v \cup \{v\}) \setminus C) \cup S_i)$ of G.

The algorithm generating the neighborhood of M is described in Algorithm 2.

▶ **Theorem 13** (★). *Algorithm 1 with `Neigborhood` in Algorithm 2 enumerates all the minimal multicuts of G in incremental polynomial time.*

Note that, in Algorithm 2, we use Takata's algorithm to enumerating minimal v-v_t separators of H'. To bound the delay of our algorithm, we need to process lines 13-14 for each output of Takata's algorithm.

4 Polynomial delay enumeration of minimal node multiway cuts

This section is devoted to designing a polynomial delay and exponential space enumeration algorithm for minimal node multiway cuts. Let $G = (V, E)$ be a graph and let T be a set of terminals. We assume hereafter that $k = |T|$. We begin with a characterization of minimal node multiway cuts as Lemma 5.

▶ **Lemma 14** (★). *A node multiway cut $M \subseteq V \setminus T$ is minimal if and only if there are k connected components $C_1, C_2, \ldots, C_k$ of $V \setminus M$ such that (1) for each $1 \leq i \leq k$, C_i contains t_i and (2) for every $v \in M$, there is a pair of indices $1 \leq i < j \leq k$ with $N(v) \cap C_i \neq \emptyset$ and $N(v) \cap C_j \neq \emptyset$.*

Algorithm 2 Computing the neighborhood of a minimal node multicut M of (G, B).

```
Function Neigborhood($M, \mathcal{M}$)
    $\mathcal{S} \leftarrow \emptyset$
    for $v \in M$ do
        for $C \in \mathcal{C}_M$ do
            $T_v \leftarrow N(v) \cap T$
            $M'' \leftarrow (M \setminus \{v\}) \cup (N(T_v \cup \{v\}) \setminus C))$
            if $G[C \cup N[v]]$ has no terminal pairs then
                if $\mu(M'') \notin \mathcal{M}$ then Output $\mu(M'')$
                $\mathcal{S} \leftarrow \mathcal{S} \cup \{\mu(M'')\}$
            else
                Run Takata's algorithm [37] for $(H', v, v_t)$ in Lemma 12
                foreach Output $M'$ of minimal $v$-$v_t$ separator in $H'$ do
                    if $\mu(M'' \cup M') \notin \mathcal{M}$ then Output $\mu(M'' \cup M')$
                    $\mathcal{S} \leftarrow \mathcal{S} \cup \{\mu(M'' \cup M')\}$
    return $\mathcal{S}$
```

From a minimal node multiway cut M of G, one can determine a set of k connected components $C_1, \ldots, C_k$ in Lemma 14. Conversely, from a set of connected components $C_1, \ldots, C_k$ satisfying (1) and (2), one can uniquely determine a minimal node multiway cut M. Given this, we denote by $\mathcal{C}_M$ a set of k connected components associated to M.

The basic strategy to enumerate minimal node multiway cuts is the same as one used in the previous section: We define a solution graph that is strongly connected. Let M be a minimal node multiway cut of G and let $\mathcal{C}_M = \{C_1, \ldots, C_k\}$. For $1 \leq i \leq k$, $v \in M$ with $N(v) \cap (T \setminus \{t_i\}) = \emptyset$, let $M^{i,v} = (M \setminus \{v\}) \cup (\bigcup_{j \neq i} N(v) \cap C_j)$. Intuitively, $M^{i,v}$ is obtained from M by moving v to C_i and then appropriately removing vertices in $N(v)$ from C_j. The key to our polynomial delay complexity is the size of the neighborhood of each M is bounded by a polynomial in n, whereas it can be exponential in the case of minimal node multicut.

▶ **Lemma 15** (★). *If M is a minimal node multiway cut of G, then so is $M^{i,v}$.*

Now, we define the neighborhood of M in the solution graph. The neighborhood of M consists of the set of minimal node multiway cuts $\mu\left(M^{i,v}\right)$ for every $1 \leq i \leq k$ and $v \in M$ with $N(v) \cap (T \setminus \{t_i\}) = \emptyset$. To show the strong connectivity of the solution graph, we define

$$\texttt{dist}(M, M') = \sum_{1 \leq i \leq k} |C'_i \setminus C_i|,$$

where $\mathcal{C}_M = \{C_1, \ldots, C_k\}$ and $\mathcal{C}_{M'} = \{C'_1, \ldots, C'_k\}$. Note that the definition of `dist` is slightly different from one used in the previous section. Each component of a node multicut may be have a several terminals. It is difficult to provide one-to-one correspondence between connected components in the two solutions. However, in a node multiway cut, each connected component has just one terminal. The fact makes it easy to provide appropriate one-to-one correspondence between connected components in the two solutions. Let M, M', M'' be minimal node multiway cuts of G. We say that M is *closer than M'' to M* if $\texttt{dist}(M, M') < \texttt{dist}(M'', M')$.

▶ **Lemma 16** (★). *Let M and M' be minimal node multiway cuts of G. Then, $\texttt{dist}(M, M') = 0$ if and only if $M = M'$.*

Algorithm 3 Computing the neighborhood of a minimal node multiway cut M of G.

```
Function Neighborhood($M, \mathcal{M}$)
    $\mathcal{S} \leftarrow \emptyset$
    for $v \in M$ do
        for $C_i \in \mathcal{C}_M$ do
            if $N(v) \setminus C_i$ has no terminals then $\mathcal{S} \leftarrow \mathcal{S} \cup \mu\left(M^{i,v}\right)$
    return $\mathcal{S}$
```

▶ **Lemma 17** (★). *Let M and M' be distinct minimal node multiway cuts of G. Then, there is a minimal node multiway cut M'' of G in the neighborhood of M such that M'' is closer than M to M'.*

Similarly to the previous section, by Lemma 17, we can conclude that the solution graph is strongly connected. From this neighborhood relation, our enumeration algorithm is quite similar to one in the previous section, which is described in Algorithm 3. To bound the delay of Algorithm 3, we need to bound the time complexity of computing $\mu(M)$.

▶ **Lemma 18** (★). *Let M be a node multiway cut of G. Then, we can compute $\mu(M)$ in $O(n + m)$ time.*

▶ **Theorem 19** (★). *Algorithm 1 with* `Neigborhood` *in Algorithm 3 enumerates all the minimal node multiway cuts of G in $O(knm)$ delay and exponential space.*

5 Polynomial space enumeration for minimal edge multiway cuts

In the previous section, we have developed a polynomial delay enumeration for both node multiway cuts. Proposition 4 and the previous result imply that the minimal edge multiway cut enumeration problem can be solved in polynomial delay and exponential space. In this section, we design a polynomial delay and space enumeration for minimal edge multiway cuts. Let $G = (V, E)$ be a graph and let T be a set of terminals.

▶ **Lemma 20** (★). *Let $M \subseteq E$ be an edge multiway cut of G. Then, M is minimal if and only if $G - M$ has exactly k connected components $C_1, \dots, C_k$, each C_i of which contains t_i.*

Note that the lemma proves in fact that there is a bijection between the set of minimal multiway cuts of G and the collection of partitions of V satisfying the condition in the lemma. In what follows, we also regard a minimal multiway cut M of G as a partition $\mathcal{P}_M = \{C_1, C_2, \dots C_k\}$ of V satisfying the condition in Lemma 20. We write $\mathcal{P}_M^{i<}$, $\mathcal{P}_M^{<i}$, and $\mathcal{P}_M^{\leq i}$ to denote $\bigcup_{i<j} C_j$, $\bigcup_{j<i} C_j$, and $\bigcup_{j\leq i} C_j$, respectively. For a vertex $v \in V$, the position of v in $\mathcal{P}_M$, denoted by $\mathcal{P}_M(v)$, is the index $1 \leq i \leq k$ with $v \in C_i$.

The bottleneck of the space complexity for enumeration algorithms in the previous sections is to use a dictionary to avoid duplication. To overcome this bottleneck, we propose an algorithm based on *the reverse search paradigm* [2]. Fix a graph $G = (V, E)$ and a terminal set $T \subseteq V$. In this paradigm, we also define a graph on the set of all minimal edge multiway cuts of G and a specific minimal edge multiway cut, which we call the *root*, denoted by $R \subseteq V$. By carefully designing the neighborhood of each minimal edge multiway cut of G, the solution graph induces a *directed tree* from the root, which enables us to enumerate those without duplication in polynomial space.

To this end, we first define the root $\mathcal{P}_R = \{C_1^r, \ldots, C_k^r\}$ as follows: Let C_i^r be the component in $G - (\mathcal{P}_R^{<i} \cup \{t_{i+1}, \ldots, t_k\})$ including t_i. Note that $\mathcal{P}_R^{<1}$ is defined as the empty set and hence C_1^r is well-defined.

▶ **Lemma 21** (★). *The root R is a minimal edge multiway cut of G.*

Next, we define the parent-child relation in the solution graph. As in the previous sections, we define a certain measure for minimal edge multiway cuts M of G: The *depth* of M as

$$\mathtt{depth}(M) = \sum_{v \in V} (\mathcal{P}_M(v) - \mathcal{P}_R(v)).$$

Intuitively, the depth of M is the sum of a "difference" of the indices of blocks in $\mathcal{P}_M$ and $\mathcal{P}_R$ that v belongs to. For two minimal edge multiway cuts M and M' of G, we say that M is *shallower than* M' if $\mathtt{depth}(M) < \mathtt{depth}(M')$. Note that the depth of M is at most kn for minimal edge multiway cut M of G. One may think that the depth of M or more specifically $\mathcal{P}_M(v) - \mathcal{P}_R(v)$ can be negative. The following two lemmas ensure that it is always non-negative.

▶ **Lemma 22** (★). *Let M be a minimal edge multiway cut of G and let $\mathcal{P}_M = \{C_1, \ldots, C_k\}$. Then, $C_i \subseteq \mathcal{P}_R^{\leq i}$ holds for every $1 \leq i \leq k$.*

▶ **Lemma 23** (★). *Let M be a minimal edge multiway cut of G and let $\mathcal{P}_M = \{C_1, \ldots, C_k\}$. Then, $\mathtt{depth}(M) = 0$ if and only if $M = R$.*

Let M be a minimal edge multiway cut of G. To ensure that the solution graph forms a tree, we define the parent of M which is shallower than M. Let $\mathcal{P}_M = \{C_1, \ldots, C_k\}$. We say that a vertex $v \in (N(C_i) \cap \mathcal{P}_M^{i<}) \setminus T$ is *shiftable into* C_i (or simply, *shiftable*). In words, a vertex is shiftable into C_i if it is non-terminal, adjacent to a vertex in C_i, and included in C_j for some $j > i$.

▶ **Lemma 24** (★). *Let M be a minimal node multiway cut of G with $M \neq R$ and let $\mathcal{P}_M = \{C_1, \ldots, C_k\}$. Then, there is at least one shiftable vertex in $V \setminus M$.*

Let $\mathcal{P}_M = \{C_1, \ldots, C_k\}$ with $M \neq R$. By Lemma 24, $V \setminus M$ has at least one shiftable vertex. The largest index i of a component C_i into which there is a shiftable vertex is denoted by $\ell(M)$. There can be more than one vertices that are shiftable into $C_{\ell(M)}$. We say that a vertex v is the *pivot* of M if v is shiftable into $C_{\ell(M)}$, and moreover, if there are more than one such vertices, we select the pivot in the following algorithmic way:

1. Let Q be the set of vertices, each of which is shiftable into $C_{\ell(M)}$.
2. If Q contains more than one vertices, we replace Q by $Q := Q \cap C_s$, where s is the maximum index with $Q \cap C_s \neq \emptyset$.
3. If Q contains more than one vertices, we compute the set of cut vertices of $G[C_s]$. If there is at least one vertex in Q that is not a cut vertex of $G[C_s]$, remove all the cut vertices of $G[C_s]$ from Q. Otherwise, that is, Q contains cut vertices only, remove a cut vertex $v \in Q$ from Q if there is another cut vertex $w \in Q$ of $G[C_s]$ such that every path between w and t_s hits v.
4. If Q contains more than one vertices, remove all but arbitrary one vertex from Q.

Note that if we apply this algorithm to Q, Q contains exactly one vertex that is shiftable into $C_{\ell(M)}$. We select the remaining vertex in Q as the pivot of M. Now, we define the *parent* of M for each $M \neq R$, denoted by $\mathtt{par}(M)$, as follows: Let $\mathcal{P}_{\mathtt{par}(M)} = \{C_1', \ldots, C_k'\}$ such that

$$C_i' = \begin{cases} C_i & (i \neq \ell(M), \mathcal{P}_M(p)) \\ C_i \cup (C_{\mathcal{P}_M(p)} \setminus C) & (i = \ell(M))) \\ C & (i = \mathcal{P}_M(p)), \end{cases}$$

Algorithm 4 Enumerating the minimal multiway cuts of G in $O(knm)$ delay and $O(kn^2)$ space.

```
1  Procedure EMC($G, M, d$)
2      if $d$ is even then Output $M$
3      for $C_i \in \mathcal{P}_M$ do
4          for $v \in B(C_i)$ with $v \neq t_i$ do
5              $\mathcal{P}' \leftarrow \mathcal{P}$                    // $\mathcal{P}' = \{C'_1, \dots, C'_k\}$
6              $C'_i \leftarrow$ the component including $t_i$ in $G[C_i \setminus \{v\}]$
7              $C \leftarrow C_i \setminus C'_i$
8              for $j$ with $j > i$ and $N(v) \cap C_j \neq \emptyset$ do
9                  $C'_j \leftarrow C_j \cup C$
10                 if par$(M') = M$ then EMC($G, M', d+1$)
11                 $C'_j \leftarrow C_j$
12     if $d$ is odd then Output $M$
```

where p is the pivot of M and C is the component in $G[C_{\mathcal{P}_M(p)} \setminus \{p\}]$ including terminal $t_{\mathcal{P}_M(p)}$. Since p has a neighbor in $C_{\ell(M)}$, $G[C'_{\ell(M)}]$ is connected, and hence $\texttt{par}(M)$ is a minimal edge multiway cut of G as well. If $M = \texttt{par}(M')$ for some minimal edge multiway cut M' of G, M' is called a *child* of M. The following lemma shows that $\texttt{par}(M)$ is shallower than M.

▶ **Lemma 25 (★).** *Let M be a minimal edge multiway cut of G with $M \neq R$. Then, $\texttt{par}(M)$ is shallower than M.*

This lemma ensures that for every minimal edge multiway cut M of G, we can eventually obtain the root R by tracing their parents at most kn times.

Finally, we are ready to design the neighborhood of each minimal edge multiway cut M of G. The neighborhood of M is defined so that it includes all the children of M and whose size is polynomial in n. Let C be a set of vertices that induces a connected subgraph in G. The *boundary of C*, denoted by $B(C)$, is the set of vertices in C that has a neighbor outside of C.

▶ **Lemma 26 (★).** *Let M and M' be minimal edge multiway cuts of G with $\texttt{par}(M') = M$. Let $\mathcal{P}_M = \{C_1, \dots, C_k\}$. Then, the pivot p of M' belongs to the boundary of $C_{\ell(M')}$ and is adjacent to a vertex in $C_{\mathcal{P}_{M'}(p)}$.*

The above lemma implies every pivot of a child of M is contained in a boundary of C_i for some $1 \leq i \leq k$. Thus, we define the neighborhood of M as follows. Let $\mathcal{P}_M = \{C_1, \dots, C_k\}$. For each C_i, we pick a vertex $v \in B(C_i)$ with $v \neq t_i$. Let C be the set of components in $G[C_i \setminus \{v\}]$ which does not include t_i. Note that C can be empty when v is not a cut vertex in $G[C_i]$. For each $1 \leq i < j \leq k$ and $N(v) \cap C_j \neq \emptyset$, $\mathcal{P}_{M'} = \{C'_1, \dots, C'_k\}$ is defined as:

$$C'_\ell = \begin{cases} C_\ell & (\ell \neq i, j) \\ C_\ell \cup (C \cup \{v\}) & (\ell = j) \\ C_\ell \setminus (C \cup \{v\}) & (\ell = i). \end{cases}$$

The neighborhood of M contains such M' if $\texttt{par}(M') = M$ for each choice of C_i, $v \in B(C_i) \setminus \{t_i\}$, and C_j. The heart of our algorithm is the following lemma.

▶ **Lemma 27 (★).** *Let M be a minimal edge multiway cut of G. Then, the neighborhood of M includes all the children of M.*

Based on Lemma 27, Algorithm 4 enumerates all the minimal edge multiway cuts of G. Finally, we analyze the delay and the space complexity of this algorithm. To bound the delay, we use the alternative output method due to Uno [39].

▶ **Theorem 28** (★). *Let G be a graph and T be a set of terminals. Algorithm 4 runs in $O(knm)$ delay and $O\big(kn^2\big)$ space, where n is the number of vertices, m is the number of edges, and k is the number of terminals.*

6 Minimal Steiner node multicuts enumeration

We have developed efficient enumeration algorithms for minimal multicuts and minimal multiway cuts so far. In this section, we consider a generalized version of node multicuts, called *Steiner node multicuts*, and discuss a relation between this problem and the minimal transversal enumeration problem on hypergraphs.

Let $G = (V, E)$ be a graph and let $T_1, T_2, \dots T_k \subseteq V$. A subset $S \subseteq V \setminus (T_1 \cup T_2 \cup \dots \cup T_k)$ is called a *Steiner node multicut* of G if for every $1 \leq i \leq k$, there is at least one pair of vertices $\{s, t\}$ in T_i such that s and t are contained in distinct components of $G - S$. If $|T_i| = 2$ for every $1 \leq i \leq k$, S is an ordinary node multicut of G. This notion was introduced by Klein et al. [27] and the problem of finding a minimum Steiner node multicut was studied in the literature [5, 27].

Let $H = (U, \mathcal{E})$ be a hypergraph. A *transversal* of H is a subset $S \subseteq U$ such that for every hyperedge $e \in \mathcal{E}$, it holds that $e \cap S \neq \emptyset$. The problem of enumerating inclusion-wise minimal transversals, also known as dualizing monotone boolean functions, is one of the most challenging problems in this field. There are several equivalent formulations of this problem and efficient enumeration algorithms developed for special hypergraphs. However, the current best enumeration algorithm for this problem is due to Fredman and Khachiyan [16], which runs in quasi-polynomial time in the size of outputs, and no output-polynomial time enumeration algorithm is known. In this section, we show that the problem of enumerating minimal Steiner node multicuts is as hard as this problem.

Let $H = (U, \mathcal{E})$ be a hypergraph. We construct a graph G and sets of terminals as follows. We begin with a clique on U. For each $e \in \mathcal{E}$, we add a pendant vertex v_e adjacent to v for each $v \in e$ and set $T_e = \{v_e : v \in e\}$. Note that G is a split graph, that is, its vertex set can be partitioned into a clique U and an independent set $\{v_e : e \in \mathcal{E}, v \in e\}$.

▶ **Lemma 29** (★). *$S \subseteq U$ is a transversal of H if and only if it is a Steiner node multicut of G.*

This lemma implies that if one can design an output-polynomial time algorithm for enumerating minimal Steiner node multicuts in a split graph, it allows us to do so for enumerating minimal transversals of hypergraphs. For the problem of enumerating minimal Steiner edge multicuts, we could neither develop an efficient algorithm nor prove some correspondence as in Lemma 29. We leave this question for future work.

References

1 S. Arora, D. Karger, and M. Karpinski. Polynomial time approximation schemes for dense instances of NP-hard problems. *Journal of Computer and System Sciences*, 58(1):193–210, 1999. doi:10.1006/jcss.1998.1605.

2 D. Avis and K. Fukuda. Reverse search for enumeration. *Discrete Applied Mathematics*, 65(1):21–46, 1996. doi:10.1016/0166-218X(95)00026-N.

3 M. Bateni, M. Hajiaghayi, P. N. Klein, and C. Mathieu. A polynomial-time approximation scheme for planar multiway cut. In *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '12, page 639–655, USA, 2012. Society for Industrial and Applied Mathematics.

4 Vincent Bouchitté and Ioan Todinca. Treewidth and minimum fill-in: Grouping the minimal separators. *SIAM J. Comput.*, 31(1):212–232, 2002. `doi:10.1137/S0097539799359683`.

5 Karl Bringmann, Danny Hermelin, Matthias Mnich, and Erik Jan [van Leeuwen]. Parameterized complexity dichotomy for Steiner Multicut. *Journal of Computer and System Sciences*, 82(6):1020–1043, 2016.

6 D. Z. Chen and X. Wu. Efficient algorithms for k-terminal cuts on planar graphs. *Algorithmica*, 38(2):299–316, February 2004. `doi:10.1007/s00453-003-1061-2`.

7 Sara Cohen, Benny Kimelfeld, and Yehoshua Sagiv. Generating all maximal induced subgraphs for hereditary and connected-hereditary graph properties. *J. Comput. Syst. Sci.*, 74(7):1147–1159, 2008. `doi:10.1016/j.jcss.2008.04.003`.

8 Alessio Conte and Takeaki Uno. New polynomial delay bounds for maximal subgraph enumeration by proximity search. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019, Phoenix, AZ, USA, June 23-26, 2019*, pages 1179–1190, 2019. `doi:10.1145/3313276.3316402`.

9 G. Călinescu, H. Karloff, and Y. Rabani. An improved approximation algorithm for multiway cut. *Journal of Computer and System Sciences*, 60(3):564–574, 2000. `doi:10.1006/jcss.1999.1687`.

10 M. Cygan, M. Pilipczuk, M. Pilipczuk, and J. O. Wojtaszczyk. On multiway cut parameterized above lower bounds. *ACM Trans. Comput. Theory*, 5(1), 2013. `doi:10.1145/2462896.2462899`.

11 E. Dahlhaus, D. S. Johnson, C. H. Papadimitriou, P. D. Seymour, and M. Yannakakis. The complexity of multiterminal cuts. *SIAM J. Comput.*, 23(4):864–894, 1994. `doi:{10.1137/S0097539792225297}`.

12 R. Diestel. *Graph Theory, 4th Edition*, volume 173 of *Graduate texts in mathematics*. Springer, 2012.

13 Jixing Feng, Xin Li, Eduardo L. Pasiliao, and John M. Shea. Jammer placement to partition wireless network. In *2014 IEEE GLOBECOM Workshops, Austin, TX, USA, December 8-12, 2014*, pages 1487–1492, 2014. `doi:10.1109/GLOCOMW.2014.7063644`.

14 L. Fireman, E. Petrank, and A. Zaks. New algorithms for simd alignment. In *Compiler Construction*, pages 1–15, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

15 Fedor V. Fomin, Ioan Todinca, and Yngve Villanger. Large induced subgraphs via triangulations and CMSO. *SIAM J. Comput.*, 44(1):54–87, 2015.

16 Michael L. Fredman and Leonid Khachiyan. On the complexity of dualization of monotone disjunctive normal forms. *J. Algorithms*, 21(3):618–628, 1996. `doi:10.1006/jagm.1996.0062`.

17 Naveen Garg, Vijay V. Vazirani, and Mihalis Yannakakis. Approximate max-flow min-(multi)cut theorems and their applications. *SIAM J. Comput.*, 25(2):235–251, 1996. `doi:10.1137/S0097539793243016`.

18 Naveen Garg, Vijay V. Vazirani, and Mihalis Yannakakis. Primal-dual approximation algorithms for integral flow and multicut in trees. *Algorithmica*, 18(1):3–20, 1997. `doi:10.1007/BF02523685`.

19 Martin Charles Golumbic. *Algorithmic Graph Theory and Perfect Graphs (Annals of Discrete Mathematics, Vol 57)*. North-Holland Publishing Co., NLD, 2004.

20 S. Guillemot. Fpt algorithms for path-transversal and cycle-transversal problems. *Discrete Optimization*, 8(1):61–71, 2011. `doi:10.1016/j.disopt.2010.05.003`.

21 Jiong Guo, Falk Hüffner, Erhan Kenar, Rolf Niedermeier, and Johannes Uhlmann. Complexity and exact algorithms for vertex multicut in interval and bounded treewidth graphs. *Eur. J. Oper. Res.*, 186(2):542–553, 2008. `doi:10.1016/j.ejor.2007.02.014`.

22 David S. Johnson, Christos H. Papadimitriou, and Mihalis Yannakakis. On generating all maximal independent sets. *Inf. Process. Lett.*, 27(3):119–123, 1988. doi:10.1016/0020-0190(88)90065-8.

23 J. H. Kappes, M. Speth, B. Andres, G. Reinelt, and C. Schnörr. Globally optimal image partitioning by multicuts. In *Proceedings of the 8th International Conference on Energy Minimization Methods in Computer Vision and Pattern Recognition*, EMMCVPR'11, page 31–44, Berlin, Heidelberg, 2011. Springer-Verlag.

24 D. R. Karger, P. Klein, C. Stein, M. Thorup, and N. E. Young. Rounding algorithms for a geometric embedding of minimum multiway cut. *Math. Oper. Res.*, 29(3):436–461, 2004. doi:10.1287/moor.1030.0086.

25 L. Khachiyan, E. Boros, K. Borys, K. Elbassioni, V. Gurvich, and K. Makino. Generating cut conjunctions in graphs and related problems. *Algorithmica*, 51(3):239–263, 2008.

26 Leonid Khachiyan, Endre Boros, Konrad Borys, Khaled M. Elbassioni, Vladimir Gurvich, and Kazuhisa Makino. Enumerating spanning and connected subsets in graphs and matroids. In *Algorithms - ESA 2006, 14th Annual European Symposium, Zurich, Switzerland, September 11-13, 2006, Proceedings*, pages 444–455, 2006. doi:10.1007/11841036_41.

27 P. N. Klein and D. Marx. Solving planar k-terminal cut in $O(n^{c\sqrt{k}})$ time. In *Proceedings of the 39th International Colloquium Conference on Automata, Languages, and Programming - Volume Part I*, ICALP'12, page 569–580, Berlin, Heidelberg, 2012. Springer-Verlag. doi:10.1007/978-3-642-31594-7_48.

28 T. Kloks and D. Kratsch. Listing all minimal separators of a graph. *SIAM J. Comput.*, 27(3):605–613, June 1998.

29 Kazuhiro Kurita and Yasuaki Kobayashi. Efficient enumerations for minimal multicuts and multiway cuts. *CoRR*, abs/2006.16222, 2020. arXiv:2006.16222.

30 D. Marx. Parameterized graph separation problems. *Theor. Comput. Sci.*, 351(3):394–406, 2006. doi:10.1016/j.tcs.2005.10.007.

31 D. Marx. A tight lower bound for planar multiway cut with fixed number of terminals. In *Proceedings of the 39th International Colloquium Conference on Automata, Languages, and Programming - Volume Part I*, ICALP'12, page 677–688, Berlin, Heidelberg, 2012. Springer-Verlag. doi:10.1007/978-3-642-31594-7_57.

32 Dániel Marx and Igor Razgon. Fixed-parameter tractability of multicut parameterized by the size of the cutset. *SIAM J. Comput.*, 43(2):355–388, 2014. doi:10.1137/110855247.

33 J. S. Provan and D. R. Shier. A paradigm for listing (s, t)-cuts in graphs. *Algorithmica*, 15(4):351–372, 1996. doi:10.1007/BF01961544.

34 Benno Schwikowski and Ewald Speckenmeyer. On enumerating all minimal solutions of feedback problems. *Discret. Appl. Math.*, 117(1-3):253–265, 2002. doi:10.1016/S0166-218X(00)00339-5.

35 H. Shen and W. Liang. Efficient enumeration of all minimal separators in a graph. *Theoretical Computer Science*, 180(1):169–180, 1997.

36 H. S. Stone. Multiprocessor scheduling with the aid of network flow algorithms. *IEEE Trans. Softw. Eng.*, 3(1):85–93, 1977. doi:10.1109/TSE.1977.233840.

37 K. Takata. Space-optimal, backtracking algorithms to list the minimal vertex separators of a graph. *Discrete Applied Mathematics*, 158(15):1660–1667, 2010.

38 S. Tsukiyama, I. Shirakawa, H. Ozaki, and H. Ariyoshi. An algorithm to enumerate all cutsets of a graph in linear time per cutset. *J. ACM*, 27(4):619–632, 1980. doi:10.1145/322217.322220.

39 T. Uno. Two general methods to reduce delay and change of enumeration algorithms. Technical report, National Institute of Informatics Technical Report E, 2003.

40 M. Xiao. Simple and improved parameterized algorithms for multiterminal cuts. *Theor. Comp. Sys.*, 46(4):723–736, 2010. doi:10.1007/s00224-009-9215-5.