Efficient enumeration of minimal multiway cuts

Kazuhiro Kurita[1,a)] Yasuaki Kobayashi[2]

Abstract: Let $G = (V, E)$ be an undirected graph and let $T \subseteq V$ be the set of terminals. A multiway cut of (G, T) is a set of edges $M \subseteq E$ that leaves each of terminals in a separate component, that is there is no path between any pair of vertices of T in $G' = (V, E \setminus M)$. In this paper, we give an efficient algorithm for enumerating inclusion-wise minimal multiway cuts of (G, T) in $O(|T| \cdot |V| \cdot |E|)$ delay and in polynomial space.

Keywords: Minimal multiway cuts, Output-sensitive enumeration, Polynomial delay, Reverse search.

1. Introduction

Let $G = (V, E)$ be an undirected graph and $T \subseteq V$. A *multiway cut* of (G, T) is a set of edges $M \subseteq E$ such that there is no path between any pair of vertices of T in $G' = (V, E \setminus M)$. The minimum multiway cut problem is of finding a smallest cardinality multiway cut of (G, T). This problem is a natural extension of the minimum *s-t* cut problem. Unfortunately, the problem is known to be NP-hard [7] even for planar graphs and for general graphs with fixed $|T| \geq 3$. Due to numerous applications (e.g. [9, 11, 18]), a lot of efforts have been devoted to solving this problem from several perspectives such as approximation algorithms [1, 5, 12], parameterized algorithms [6, 10, 15, 21], and restricting input [3, 4, 7, 14, 16].

In this paper, we tackle this problem from yet another view point, which we focus on *enumeration*. Since the problem of finding a *minimum* multiway cut is intractable, we rather enumerates *minimal* multiway cuts. We say that a multiway cut M of $(G = (V, E), T)$ is minimal if M' is not a multiway cut of (G, T) for every proper subset $M' \subset M$. Since finding a minimal multiway cut is easy, the goal of our problem is to find all the minimal multiway cuts of a given graph G and T. In this context, there are several results related to our problem. For example, there are linear delay algorithms for enumerating all minimal *s-t* cuts [17,19], which is a special case of our problem, where T contains exactly two vertices s and t. Here, an enumeration algorithm has *delay* $f(n)$ 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 input.

Khachiyan et al. [13] studied a highly related problem, called the minimum *multicut* problem, in the context of enumeration. Given an undirected graph $G = (V, E)$ and a set of pairs of $S \subseteq V \times V$, the goal of this problem to enumerate all minimal S-cuts M of G, that is, $G' = (V, E \setminus M)$ has no paths between every vertex pair in S and M is inclusion-wise minimal with respect to this property. The cut M is called a *multicut* of (G, S). Our problem is indeed a special case of this problem, where $S = T \times T$. They gave an efficient algorithm for this problem. Their algorithm runs in *incremental polynomial time*, that is, if $\mathcal{M}$ is the set of minimal S-cuts of (G, S) that are generated so far, then the algorithm decides whether there is a minimal S-cut of G not included in $\mathcal{M}$ in time polynomial in $|V| + |E| + |\mathcal{M}|$. Moreover, if such a minimal S-cut exists, the algorithm outputs one of them within the same running time bound. Since our problem is a special case of their problem, there algorithm works for our problem as well. However, there can be exponentially many minimal multicuts in a graph. Therefore, the delay of their algorithm cannot be upper bounded by a polynomial in terms of input size.

In this paper, we give a polynomial delay algorithm for enumerating minimal multiway cut of $(G = (V, E), T)$.

Theorem 1. *There is an algorithm enumerates all the minimal multiway cuts of G in $O(|T| \cdot |V| \cdot |E|)$ delay.*

This simultaneously improves the previous incremental polynomial running time bound obtained by applying the algorithm of [13] to our problem and extends enumeration algorithms for minimal *s-t* cuts due to [17, 19]. The idea behind this result is that we rather enumerate particular partitions of V than directly enumerating minimal multiway cuts of (G, T). It is known that an *s-t* 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 [8]. This fact is highly exploited in enumerating minimal *s-t* cuts [17, 19] and can be extended for multiway cuts (See Section 2). To enumerate such partitions of V, we use the *reverse search paradigm* due to Avis and Fukuda [2]. It would be worth mentioning that the approaches for enumerating minimal *s-t* cuts of [17, 19] rely on the fact that T contains exactly two vertices, and hence it would be non-trivial to extend their approaches to the general case where $|T| > 2$.

As a straightforward application of Theorem 1, we can enumerate minimal multicuts in polynomial delay for a special case. For an instance (G, S) of the multicut problem, the *demand graph*

[1] Hokkaido University
[2] Kyoto University
[a)] k-kurita@ist.hokudai.ac.jp

is an undirected simple graph $H = (T, E_{st})$ such that T is the set of vertices appeared in S, that is, $T = \bigcup_{(s,t)\in S}\{s,t\}$, and $E_{st} = \{\{s,t\} : (s,t) \in S\}$. Note that for multiway cuts, H is exactly the complete graph on T. Provan and Shier [17] implicitly proved that if H is a star, then all the minimal multicuts can be enumerated in polynomial delay since such a minimal multicut forms a bipartition $(X, V \setminus X)$ of V such that X contains the center of H and $V \setminus X$ contains the other leaves of H (See [17], for details). The only remaining case for demand graphs of three vertices is K_3, which corresponds to our case. As an immediate consequence from Theorem 1, we have the following.

Corollary 2. *There is an algorithm enumerates all the minimal multicuts of G in polynomial delay, provided that the demand graph has at most three vertices.*

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 $N(X)$ (i.e. $N(X) = \{y \in V \setminus X : x \in X \wedge \{x, y\} \in E\}$. For a set of edges $F \subseteq E$, the graph obtained from G by deleting F is denoted by $G - F$.

Let $T = \{t_1, t_2, \ldots, t_k\}$ be a set of vertices in V. A vertex $t_i \in T$ is called a *terminal* and T is called a *terminal set or terminals*. A set of edges $M \subseteq E$ is a *multiway cut* of (G, T) if no pair of terminals is connected in $G - M$. When T is clear from the context, we simply call M a multiway cut of G. A multiway cut M is *minimal* if $M \setminus F$ is not a multiway cut of G for any $F \subset M$. Note that this condition is equivalent to that $M \setminus \{e\}$ is not a multiway cut of G for any $e \in M$. In the following lemma, we give a characterization of a minimal multiway cut of G.

Lemma 3. *Let $M \subseteq E$ be a multiway cut of G. Then, M is minimal if and only if it satisfies the following two conditions: (I) $G - M$ has k connected components $C_1, \ldots, C_k$, (II) the i-th connected component C_i is an induced subgraph of G and contains t_i.*

Proof. Suppose that M is a minimal multiway cut of G. From the definition of a minimal multiway cut, $G - M$ has at least k connected components $C_1, C_2, \ldots, C_{k'}$. We can assume without loss of generality that each C_i contains t_i. If C_i is not an induced subgraph of G, we can simply remove an edge of M whose endpoints belong to C_i, which contradicts to the minimality of M. Moreover, if $k' > k$, there is at least one edge in M whose one endpoint belongs to C_i for some $i \leq k$ and the other endpoint belongs to C_j for some $j > k$. This edge is also removable from M, contradicting to the minimality of M. Therefore, the "only if" part follows.

Conversely, let $C_1, C_2, \ldots C_k$ be the connected components of $G - M$ that satisfies (I) and (II). Since every C_i is an induced subgraph of G, every edge e in M lies between two connected components, say C_i and C_j. This implies that there is a path between t_i and t_j in $G - (M \setminus \{e\})$. Hence, M is minimal. □

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 (I) and (II). Therefore, in what follows, we regard a minimal multiway cut M as a partition $\mathcal{P} = \{C_1, C_2, \ldots C_k\}$ of V satisfying the conditions (I) and (II) in Lemma 3. In this context, we call a partition of V satisfying (I) and (II) in Lemma 3 a *valid partition*. We write $\mathcal{P}_{i<}$ and $\mathcal{P}_{<i}$ to denote $\bigcup_{i<j} C_j$ and $\bigcup_{j<i} C_j$, respectively. For a vertex $v \in V$ and a valid partition $\mathcal{P}$, the *position of v in $\mathcal{P}$*, denoted by $\mathcal{P}(v)$, is the index $1 \leq i \leq k$ with $v \in C_i$.

3. The algorithm

Fix a graph $G = (V, E)$ and a terminal set $T \subseteq V$. Our proposed algorithm follows *the reverse search paradigm* due to [2]. In this paradigm, algorithms inductively enumerate solutions from a special solution. For our problem, we inductively define this special solution $\mathcal{R} = \left\{C_1^r, \ldots, C_k^r\right\}$ as follows: Let C_i^r be the component in $G[V \setminus (\mathcal{R}_{<i} \cup \{t_{i+1}, \ldots, t_k\})]$ including t_i. Note that $\mathcal{R}_{<1}$ is defined as the empty set and hence C_1^r is well-defined.

Lemma 4. *$\mathcal{R}$ is a valid partition.*

Proof. Clearly, C_i^r satisfies (II) in Lemma 3 for all $1 \leq i \leq k$. Thus, we show that $\mathcal{R}$ is a partition of V. Let v be an arbitrary vertex of G. Since G is connected, we can assume that v is adjacent to a vertex in C_i^r for some $1 \leq i \leq k$. This implies that v is included in C_j^r for some $j \leq i$. □

This partition $\mathcal{R}$ is called the *root partition*.

Lemma 5. *Let $\mathcal{P} = \{C_1, \ldots, C_k\}$ be an arbitrary valid partition. Then, $C_i \cap C_j^r = \emptyset$ holds for every $1 \leq i < j \leq k$.*

Proof. Suppose for contradiction that v is a vertex in $C_i \cap C_j^r$ for some i and j with $i < j$. Assume, without loss of generality, there is no component $C_{j'}^r$ satisfying $C_i \cap C_{j'}^r \neq \emptyset$ with $j' > j$. Let P be a path in $G[C_i]$ between t_i and v. Assume moreover that there is no vertex in C_j^r on the path other than v. From these assumptions, for every w on P except for v, we have $\mathcal{R}(w) < j$. Let w be the vertex on P adjacent to v. By the definition of $\mathcal{R}$, both w and v are included in the same component $C_{\mathcal{R}(w)}^r$, which contradicts to the fact that $\mathcal{R}(w) < j$. □

Corollary 6. *Let $\mathcal{P} = \{C_1, \ldots, C_k\}$ be an arbitrary valid partition. Then, $C_i \subseteq \mathcal{R}_{\leq i}$ holds for every $1 \leq i \leq k$.*

We define the *farness* of $\mathcal{P}$ as:

$$\sum_{v\in V}(\mathcal{P}(v) - \mathcal{R}(v)).$$

Intuitively, the farness of a valid partition is the sum of the "difference" of the indices of v in $\mathcal{P}$ and in $\mathcal{R}$. Note that the farness of $\mathcal{P}$ is at most kn for every valid partition $\mathcal{P}$. One may think that the farness or more specifically $\mathcal{P}(v) - \mathcal{R}(v)$ can be negative. The following lemma ensures that it is always non-negative.

Lemma 7. *Let $\mathcal{P} = \{C_1, \ldots, C_k\}$ be a valid partition. Then, the farness of $\mathcal{P}$ is equal to zero if and only if $\mathcal{P}$ is the root partition.*

Proof. Obviously, the farness of the root partition $\mathcal{R}$ is zero. Thus, in the following, we consider the "only if" part. By Corollary 6, every vertex $v \in C_i$ is included in $\mathcal{R}_{\leq i}$. This implies that $\mathcal{P}(v) - \mathcal{R}(v)$ is non-negative. Since the farness of $\mathcal{P}$ is equal to zero, we have $\mathcal{P}(v) = \mathcal{R}(v)$ for every $v \in V$. Hence, we have $C_i = C_i^r$ for every $1 \leq i \leq k$. □

Let $\mathcal{P} = \{C_1, \ldots, C_k\}$ be a valid partition. We say that a vertex $v \in (N(C_i) \cap \mathcal{P}_{i<}) \setminus T$ 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 $j > i$ (See Figure 1).

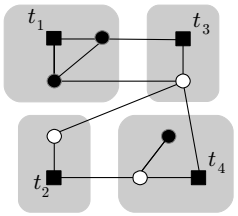

Fig. 1 The figure illustrates a valid partition of a graph with four terminals. Squares indicate terminals and circle indicate other vertices. Shiftable vertices are depicted by white circles.

Lemma 8. *Let $\mathcal{P}$ be a valid partition with $\mathcal{P} \neq \mathcal{R}$. Then, $\mathcal{P}$ has at least one shiftable vertex.*

Proof. By Lemma 7, the farness of $\mathcal{P}$ is more than zero. This implies that there is a vertex $v \in C_j \cap C_i^r \neq \emptyset$ for some $i \neq j$. Note that v is not a terminal. By Lemma 5, we have $i < j$. Observe that $C_i^r \setminus C_j$ is not empty since C_i^r contains terminal t_i that is not contained in C_j. Since $G[C_i^r]$ is connected, there is at least one vertex $w \in C_i^r \setminus C_j$ that is adjacent to v. Note that w is not included in C_j, which implies that $j \neq \mathcal{P}(w)$. If $j < \mathcal{P}(w)$, we have $w \neq t_i$ and hence w is shiftable into C_j. Otherwise, $j > \mathcal{P}(w)$, we can conclude that v is shiftable into $C_{\mathcal{P}(w)}$. Hence the lemma follows. □

Let $\mathcal{P} = \{C_1, \ldots, C_k\}$ be a valid partition with $\mathcal{P} \neq \mathcal{R}$. By Lemma 8, $\mathcal{P}$ 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 $\ell(\mathcal{P})$. There can be more than one vertices that are shiftable into $C_{\ell(\mathcal{P})}$. We say that a vertex v is the *pivot* of $\mathcal{P}$ if v is shiftable into $C_{\ell(\mathcal{P})}$, 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 $C_{\ell(\mathcal{P})}$.
(2) If Q contains more than one vertices, we replace Q as $Q := Q \cap C_s$, where s is the maximum index with $Q \cap C_s \neq \emptyset$.
(3) If Q contains more than one vertices, we compute the cut vertices in $G[C_s]$. 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 only cut vertices of $G[C_s]$, 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 one vertex that is the minimum with respect to some prescribed total order on V.

Note that if we apply this algorithm to Q, Q contains exactly one vertex that is shiftable into $C_{\ell(\mathcal{P})}$. We select this vertex as the pivot of $\mathcal{P}$.

Lemma 9. *Let p be the pivot of $\mathcal{P} = \{C_1, \ldots C_k\}$ and let $s = \mathcal{P}(p)$. Then, for every connected component C of $G[C_s \setminus \{p\}]$, either C contains terminal t_s or C has no any shiftable vertex into $C_{\ell(\mathcal{P})}$.*

Proof. If p is not a cut vertex in $G[C_s]$, clearly $G[C_s \setminus \{p\}]$ has exactly one component that has terminal t_s. Suppose otherwise. If there is a component C of $G[C_s \setminus \{p\}]$ that has no terminal t_s and has a shiftable vertex v into $C_{\ell(\mathcal{P})}$. Then, since p is the cut vertex of $G[C_s]$, every path between v and t_s hits p. This contradicts to the choice of p. □

To completely enumerate all the valid partitions from the root partition, we define a tree whose node corresponds to a valid partition and the root corresponds to the root partition. Once we can define this tree structure on the set of valid partitions, we can enumerate all the valid partitions from the root partition without duplication. To this end, we define the *parent* of a valid partition $\mathcal{P} = \{C_1, C_2, \ldots, C_k\}$ with $\mathcal{P} \neq \mathcal{R}$, denoted by $\mathtt{par}(\mathcal{P}) = \{C'_1, \ldots, C'_k\}$, as follows:

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

where p is the pivot of $\mathcal{P}$ and C is the component in $G[C_{\mathcal{P}(p)} \setminus \{p\}]$ including terminal $t_{\mathcal{P}(p)}$. Since p has a neighbor in $C_{\ell(\mathcal{P})}$, $G[C'_{\ell(P)}]$ is connected, and hence $\mathtt{par}(\mathcal{P})$ is a valid partition as well. If $\mathcal{P} = \mathtt{par}(\mathcal{P}')$ for some $\mathcal{P}'$, $\mathcal{P}'$ is called a *child* of $\mathcal{P}$.

Observe that for every valid partition $\mathcal{P}$ that is not the root partition $\mathcal{R}$, the farness of $\mathtt{par}(\mathcal{P})$ is strictly smaller than that of $\mathcal{P}$ since $\mathtt{par}(\mathcal{P})(v) \leq \mathcal{P}(v)$ for every $v \in V$ and $\mathtt{par}(\mathcal{P})(p) < \mathcal{P}(p)$ holds for the pivot p of $\mathcal{P}$. This infers that for every valid partition $\mathcal{P}$ with $\mathcal{P} \neq \mathcal{R}$, we can eventually obtain $\mathcal{R}$ by tracing their parents at most kn times. Therefore, this parent-children relation defines a tree whose root is the root partition and contains all the valid partitions as its nodes.

From a valid partition that is not the root partition, we can easily find its parent. However, to traverse the tree to enumerate the valid partitions, we need to compute the set of children from their parent. For a valid partition $\mathcal{P}$, we denote by $\mathtt{ch}(\mathcal{P})$ the set of all children of $\mathcal{P}$. 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 10. *Let $\mathcal{P} = \{C_1, \ldots, C_k\}$ be a valid partition and $\mathcal{P}'$ a child of $\mathcal{P}$. Then, the pivot p of $\mathcal{P}'$ belongs to the boundary of $C_{\ell(\mathcal{P}')}$ and is adjacent to a vertex in $C_{\mathcal{P}'(p)}$.*

Proof. Let $\mathcal{P}' = \{C'_1, \ldots, C'_k\}$ and let $s = \mathcal{P}'(p)$. Since p is shiftable, it belongs to the boundary of C'_s. Moreover, p belongs to $C_{\ell(\mathcal{P}')}$. Since $G[C'_s]$ is connected and has at least two vertices (p and t_s), p has a neighbor w in C'_s. We can choose w as a vertex in the component of $G[C'_s \setminus \{p\}]$ including terminal t_s. This implies that $\mathcal{P}(w) = s$ and hence p belongs to the boundary of $C_{\ell(\mathcal{P}')}$ and is adjacent to $w \in C_s$. □

Our algorithm `EMC` is described in Algorithm 1. For enumerating valid partitions, call `EMC` $(G, \mathcal{R}, 0)$, where $\mathcal{R}$ is the root partition for G.

Lemma 11. *Given the root partition $\mathcal{R}$ for G, Algorithm 1 enumerates all the valid partitions for G.*

Proof. To prove the correctness of Algorithm 1, it suffices to

Algorithm 1: An algorithm to enumerate minimal multiway cuts in $O(knm)$ delay and $O\left(kn^2+m\right)$ space.

```
1  Procedure EMC(G, P = {C_1, ..., C_k}, d)
2      if d is even then Output P
3      for C_i ∈ P do
4          for v ∈ B(C_i) with v ≠ t_i do
5              ID ← ∅
6              for u ∈ N(v) \ C_i do ID ← ID ∪ {P(u)}
7              P' ← P                          // P' = {C'_1, ..., C'_k}
8              C'_i ← the component including t_i in G[C_i \ {v}]
9              C ← C_i \ C'_i
10             for j ∈ ID with j > i do
11                 C'_j ← C_j ∪ C
12                 if par(P') = P then EMC(G, P', d + 1)
13                 C'_j ← C_j
14     if d is odd then Output P
```

show that given a valid partition $\mathcal{P} = \{C_1, \ldots, C_k\}$, every child of $\mathcal{P}$ is generated in line 3 to 13 in EMC. Basically, EMC tries to find a child $\mathcal{P}' = \left\{C'_1, \ldots, C'_k\right\}$ of a given valid partition $\mathcal{P} = \{C_1, \ldots, C_k\}$. For a valid partition $\mathcal{P}$ and its child $\mathcal{P}'$, every component C_i except two is equal to the corresponding component C'_i. The only difference between them is two pairs of components $(C_{\ell(\mathcal{P}')}, C'_{\ell(\mathcal{P}')})$ and $(C_{\mathcal{P}'(p)}, C'_{\mathcal{P}'(p)})$. Line 3 guesses the component $C_{\ell(\mathcal{P}')}$. By Lemma 10, the pivot of $\mathcal{P}'$ is included in the boundary of $C_{\ell(\mathcal{P}')}$. Thus, line 4 correctly guesses the pivot p of $\mathcal{P}'$. Moreover, since, by Lemma 10, p has a neighbor in $C'_{\mathcal{P}'(p)}$, lines 10-13 correctly guess the index $\mathcal{P}'(p)$.

Now, consider two components $C_{\ell(\mathcal{P}')}$ and $C_{\mathcal{P}'(p)}$. Let $\ell = \ell(\mathcal{P}')$ and $s = \mathcal{P}'(p)$. By the definition of a parent, $C_\ell = C'_\ell \cup (C'_s \setminus C_s)$ and C_s is the component of $G[C'_s \setminus \{p\}]$ including terminal t_s. Since $C'_\ell \cap C'_s = \emptyset$ and $C_\ell \cap C_s = \emptyset$, we have $C'_\ell = C_\ell \setminus (C'_s \setminus C_s)$. By Lemma 9, $C'_s \setminus C_s$ has only one shiftable vertex into C'_ℓ, which is the pivot p of $\mathcal{P}$. By the definition of shiftable vertex, there are no edges between a vertex in $C'_s \setminus C_s \setminus \{p\}$ and a vertex in C'_l. This means that either $C'_s \setminus C_s \setminus \{p\}$ is empty or p is a cut vertex in $G[C_\ell \setminus \{p\}]$ that separates $C'_s \setminus C_s \setminus \{p\}$ from C'_ℓ. Therefore, C'_ℓ is the component of $G[C'_l \setminus \{p\}]$ including terminal t_ℓ. Moreover, by the definition of a parent, we have $C'_s = C_s \cup (C_\ell \setminus C'_\ell)$.

Hence, Algorithm 1 correctly computes all the valid partitions for G. □

Lemma 12. *Given the root partition $\mathcal{R}$ for G, Algorithm 1 runs in $O(knm)$ delay and $O\left(kn^2+m\right)$ space.*

Proof. Let $\mathcal{T}$ be the enumeration tree defined by the recursive calls of EMC.

First, we analyze the total running time of EMC. Let $\mathcal{P} = \{C_1, \ldots, C_k\}$ be a valid partition. In each node of $\mathcal{T}$, line 3 guesses the component $C_i \in \mathcal{P}$ and line 4 guesses the vertex in the boundary $B(C_i)$. The loop procedure from line 4 to line 13 is executed at most n times since the total size of boundaries is at most n. Since the computation of $\mathcal{P}'$ and $\mathtt{par}(\mathcal{P}')$ can be done in $O(m)$ time for each C_j, each node needs $O(knm)$ time. Since the algorithm outputs exactly one valid partition in each node of $\mathcal{T}$, the total computation time is $O(Mknm)$, where M is the number of valid partitions.

Next, we analyze the delay of EMC. To show a delay bound, we use the alternative output method due to [20]. We replace each edge of $\mathcal{T}$ with a pair of parallel edges. Then, the traversal of $\mathcal{T}$ naturally defines an Eulerian tour on this replaced graph. Let $S = (n_1, \ldots, n_t)$ be the sequence of nodes that appear on this tour in this order. Note that each leaf node appears exactly once in S and each internal node appears more than once in S. From now on, we may call each n_i an *event* and denote by e_i the i-th event in S. Observe that if the depth of n_i is even (resp. odd) in $\mathcal{T}$, then the first (resp. last) event in S corresponding to this node outputs a valid partition. Now, let us consider three consecutive events e_i, e_{i+1}, and e_{i+2} in S. Since each node is processed in $O(knm)$ time, it suffices to show that at least one of these events outputs a valid partition. If at least one of these events corresponds to a leaf node, this claim obviously holds. Hence, we assume not in this case. Since each of e_i, e_{i+1}, and e_{i+2} corresponds to an internal node of $\mathcal{T}$, there are three possibilities (Figure 2): (I) n_{i+1} is a child of n_i and n_{i+2} is a child of n_{i+1}. (II) n_{i+1} is a parent of n_i and n_{i+2} is a parent of n_{i+1}. (III) n_{i+1} is a parent of both n_i and n_{i+2}. Note that these three nodes must be distinct since none of them is a leaf of $\mathcal{T}$. For case (I), the events e_{i+1} and e_{i+2} are the first events for distinct nodes n_{i+1} and n_{i+2}, respectively. Since exactly one of n_{i+1} and n_{i+2} has even depth, therefore, either e_{i+1} or e_{i+2} outputs a valid partition. For case (II), the events e_{i+1} and e_{i+2} are the last events for those nodes, and hence exactly one of them outputs a valid partition as well. For case (III), suppose first that n_{i+1} has even depth. Then n_i has odd depth, and hence e_i is the last event for n_i and hence e_i outputs a valid partition. Suppose otherwise that n_{i+1} has odd depth. Then, n_{i+2} has even depth and e_{i+2} is the first event for this node. This, e_{i+2} outputs a valid partition. Therefore, the delay of EMC is $O(knm)$.

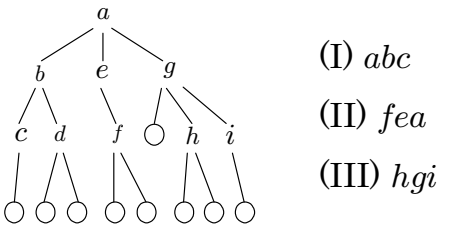

Fig. 2 The figure depicts the three cases of consecutive three events e_i, e_{i+1}, and e_{i+2}.

Finally, we analyze the space complexity. Let $\mathcal{P}$ be a solution and P be a path between $\mathcal{P}$ and $\mathcal{R}$ in $\mathcal{T}$. In each node, we store the solution $\mathcal{P}'$, the boundary for each $C'_i \in \mathcal{P}'$ and a set of indices ID. Since the size of each set and partition is $O(n)$ and the depth of $\mathcal{T}$ is kn, the space complexity is $O\left(kn^2+m\right)$. Hence, the statement holds. □

References

[1] Arora, S., Karger, D. and Karpinski, M.: Polynomial Time Approximation Schemes for Dense Instances of NP-Hard Problems, *Journal of Computer and System Sciences*, Vol. 58, No. 1, pp. 193 – 210 (online), DOI: https://doi.org/10.1006/jcss.1998.1605 (1999).

[2] Avis, D. and Fukuda, K.: Reverse search for enumeration, *Discrete Applied Mathematics*, Vol. 65, No. 1, pp. 21 – 46 (online), DOI: https://doi.org/10.1016/0166-218X(95)00026-N (1996).

[3] Bateni, M., Hajiaghayi, M., Klein, P. N. and Mathieu, C.: A Polynomial-Time Approximation Scheme for Planar Multiway Cut,

Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA '12, USA, Society for Industrial and Applied Mathematics, p. 639–655 (2012).

[4] Chen, D. Z. and Wu, X.: Efficient Algorithms for k-Terminal Cuts on Planar Graphs, *Algorithmica*, Vol. 38, No. 2, pp. 299–316 (online), DOI: 10.1007/s00453-003-1061-2 (2004).

[5] Călinescu, G., Karloff, H. and Rabani, Y.: An Improved Approximation Algorithm for MULTIWAY CUT, *Journal of Computer and System Sciences*, Vol. 60, No. 3, pp. 564 – 574 (online), DOI: https://doi.org/10.1006/jcss.1999.1687 (2000).

[6] Cygan, M., Pilipczuk, M., Pilipczuk, M. and Wojtaszczyk, J. O.: On Multiway Cut Parameterized above Lower Bounds, *ACM Trans. Comput. Theory*, Vol. 5, No. 1 (online), DOI: 10.1145/2462896.2462899 (2013).

[7] Dahlhaus, E., Johnson, D. S., Papadimitriou, C. H., Seymour, P. D. and Yannakakis, M.: The Complexity of Multiterminal Cuts, *SIAM J. Comput.*, Vol. 23, No. 4, p. 864–894 (online), DOI: 10.1137/S0097539792225297 (1994).

[8] Diestel, R.: *Graph Theory, 4th Edition*, Graduate texts in mathematics, Vol. 173, Springer (2012).

[9] Fireman, L., Petrank, E. and Zaks, A.: New Algorithms for SIMD Alignment, *Compiler Construction*, Berlin, Heidelberg, Springer Berlin Heidelberg, pp. 1–15 (2007).

[10] Guillemot, S.: FPT algorithms for path-transversal and cycle-transversal problems, *Discrete Optimization*, Vol. 8, No. 1, pp. 61–71 (online), DOI: https://doi.org/10.1016/j.disopt.2010.05.003 (2011).

[11] Kappes, J. H., Speth, M., Andres, B., Reinelt, G. and Schnörr, C.: Globally Optimal Image Partitioning by Multicuts, *Proceedings of the 8th International Conference on Energy Minimization Methods in Computer Vision and Pattern Recognition*, EMMCVPR '11, Berlin, Heidelberg, Springer-Verlag, p. 31–44 (2011).

[12] Karger, D. R., Klein, P., Stein, C., Thorup, M. and Young, N. E.: Rounding Algorithms for a Geometric Embedding of Minimum Multiway Cut, *Math. Oper. Res.*, Vol. 29, No. 3, p. 436–461 (online), DOI: 10.1287/moor.1030.0086 (2004).

[13] Khachiyan, L., Boros, E., Borys, K., Elbassioni, K., Gurvich, V. and Makino, K.: Generating Cut Conjunctions in Graphs and Related Problems, *Algorithmica*, Vol. 51, No. 3, p. 239–263 (2008).

[14] Klein, P. N. and Marx, D.: Solving Planar K-Terminal Cut in $O(N^{c\sqrt{k}})$ Time, *Proceedings of the 39th International Colloquium Conference on Automata, Languages, and Programming - Volume Part I*, ICALP'12, Berlin, Heidelberg, Springer-Verlag, p. 569–580 (online), DOI: 10.1007/978-3-642-31594-7_48 (2012).

[15] Marx, D.: Parameterized Graph Separation Problems, *Theor. Comput. Sci.*, Vol. 351, No. 3, p. 394–406 (online), DOI: 10.1016/j.tcs.2005.10.007 (2006).

[16] Marx, D.: A Tight Lower Bound for Planar Multiway Cut with Fixed Number of Terminals, *Proceedings of the 39th International Colloquium Conference on Automata, Languages, and Programming - Volume Part I*, ICALP '12, Berlin, Heidelberg, Springer-Verlag, p. 677–688 (online), DOI: 10.1007/978-3-642-31594-7_57 (2012).

[17] Provan, J. S. and Shier, D. R.: A Paradigm for Listing (s,t)-Cuts in Graphs, *Algorithmica*, Vol. 15, No. 4, p. 351–372 (online), DOI: 10.1007/BF01961544 (1996).

[18] Stone, H. S.: Multiprocessor Scheduling with the Aid of Network Flow Algorithms, *IEEE Trans. Softw. Eng.*, Vol. 3, No. 1, p. 85–93 (online), DOI: 10.1109/TSE.1977.233840 (1977).

[19] Tsukiyama, S., Shirakawa, I., Ozaki, H. and Ariyoshi, H.: An Algorithm to Enumerate All Cutsets of a Graph in Linear Time per Cutset, *J. ACM*, Vol. 27, No. 4, p. 619–632 (online), DOI: 10.1145/322217.322220 (1980).

[20] Uno, T.: Two general methods to reduce delay and change of enumeration algorithms, Technical report, National Institute of Informatics Technical Report E (2003).

[21] Xiao, M.: Simple and Improved Parameterized Algorithms for Multiterminal Cuts, *Theor. Comp. Sys.*, Vol. 46, No. 4, p. 723–736 (online), DOI: 10.1007/s00224-009-9215-5 (2010).