Computing Minimum Multiway Cuts in Hypergraphs from Hypertree Packings*

Takuro Fukunaga

Department of Applied Mathematics and Physics,
Graduate School of Informatics, Kyoto University
`takuro@amp.i.kyoto-u.ac.jp`

Abstract

Hypergraph multiway cut problem is a problem of finding a minimum capacity set of hyperedges whose removal divides a given hypergraph into a specified number of connected components. We present an algorithm for this problem which runs in strongly polynomial-time if both the specified number of connected components and the maximum size of hyperedges in the hypergraph are constants. Our algorithm extends the algorithm due to Thorup (2008) for computing minimum multiway cuts of graphs from greedy packings of spanning trees.

1 Introduction

For a connected hypergraph $H = (V, E)$ with a non-negative hyperedge capacity $c : E \rightarrow \mathbb{Q}_+$ and an integer $k \geq 2$, a k -cut of H is defined as a subset of E whose removal partitions H into k connected components. In this paper, we consider the *hypergraph multiway cut problem*, which is a problem of finding a minimum capacity k -cut of a hypergraph.

The graph multiway cut problem is one of the fundamental problems in combinatorial optimization. It is closely related to the reliability of networks, and has many applications, for example, to the traveling salesperson problem, VLSI design, and evolutionary tree construction [3, 9]. By Goldschmidt and Hochbaum [5], it is shown that the problem is NP-hard when k is not fixed, and polynomial-time solvable when k is fixed to a constant. After their work, there are many works on the algorithmic aspect of this problem.

In spite of these active studies on the graph multiway cut problem, there are few works on the hypergraph multiway cut problem. If k is not fixed, the NP-hardness of the graph multiway cut problem implies that of the hypergraph multiway cut problem. For the case of $k = 3$, Xiao [13] gave a polynomial-time algorithm. However, we do not know whether the hypergraph multiway cut problem is polynomial solvable or NP-hard when k is a constant larger than 3.

The contribution of this paper is to answer this question. We present an algorithm which is strongly polynomial-time if k and the maximum size γ of hyperedges (i.e., $\gamma = \max_{e \in E} |e|$) are fixed to constants. Since graphs can be regarded as hypergraphs with $\gamma = 2$, this result extends the polynomial-solvability of the graph multiway cut problem.

*Technical Report 2009-016, August 5, 2009.

Our algorithm is based on the idea due to Thorup [12], which is successfully applied to the graph multiway cut problem. He showed that a maximum spanning tree packing of a graph contains a spanning tree sharing at most a constant number of edges with a minimum k -cut of the graph. Although this fact itself gives a strongly polynomial-time algorithm for computing the minimum k -cuts of graphs, he also showed that a set of spanning trees constructed in some greedy way has the same property. Based on this fact, he gave an algorithm, which is fastest so far. In this paper, we show that these facts can be extended to hypergraphs with a hypertree packing theorem due to Frank, Király and Kriesell [1] (see Section 3).

Let us mention the previous works on problems related to the hypergraph multiway cut problem. As mentioned above, the first polynomial-time algorithm for the graph multiway cut problem with fixed k was presented by Goldschmidt and Hochbaum [5]. Its running time is $O(n^{k^2}T(n, m))$ where $T(n, m)$ is time for computing max-flow in a graph consisting of n vertices and m edges. $T(n, m)$ is known to be $O(mn \log(n^2/m))$ for now [4]. After their work, many polynomial-time algorithms for fixed k are obtained. An algorithm due to Kamidoi, Yoshida and Nagamochi [6] runs in $O(n^{4k/(1-1.71/\sqrt{k})-34}T(n, m))$. An algorithm due to Xiao [14] runs in $O(n^{4k-\log k})$. An algorithm due to Thorup [11] runs in $\tilde{O}(n^{2k})$. In addition, Karger and Stein [7] gave a randomized algorithm running in $O(n^{2(k-1)} \log^3 n)$.

For the hypergraph multiway cut problem, Xiao [13] gave a polynomial-time divide-and-conquer algorithm for $k = 3$. Zhao, Nagamochi and Ibaraki [15] gave an approximation algorithm. It achieves the approximation factor $(1 - \frac{2}{k}) \min\{k, \gamma\}$ for $k \geq 4$ by using the algorithm due to Xiao for $k = 3$. Moreover, it is shown by Okumoto, Fukunaga and Nagamochi [10] that the problem can be reduced to the terminal k -vertex cut problem in bipartite graphs. Hence the LP-rounding algorithm due to Garg, Vazirani and Yannakakis [2] for the terminal k -vertex cut problem achieves approximation factor $2 - \frac{2}{k}$ also for the hypergraph multiway cut problem.

Okumoto, Fukunaga and Nagamochi [10] showed that the hypergraph k -cut problem is contained by the submodular k -partition problem with asymmetric submodular functions. Zhao, Nagamochi and Ibaraki [15] presented a $(k - 1)$ -approximation algorithm to this problem. Okumoto, Fukunaga and Nagamochi [10] presented an approximation algorithm whose approximation factor is 1.5 for $k = 4$ and $k + 1 - 2\sqrt{k-1}$ for $k \geq 5$. They also showed that, for the hypergraph multiway cut problem with $k = 4$, their algorithm achieves approximation factor $4/3$.

The rest of this paper is organized as follows. Section 2 introduces basic facts and notations. Section 3 explains outline of our result and presents our algorithm. Section 4 shows that a maximum hypertree packing contains a hypertree sharing at most a constant number of hyperedges with a minimum k -cut. Section 5 discusses a property of a set of hypertrees constructed greedily. Section 6 concludes this paper and mentions the future works.

2 Preliminaries

$\mathbb{Q}_+$ denotes the set of non-negative rationals. Let $H = (V, E)$ be a hypergraph with a capacity $c : E \rightarrow \mathbb{Q}_+$. Throughout this paper, we denote $|V|$ by n , $|E|$ by m , and $\max_{e \in E} |e|$ by γ . We sometimes denote the vertex set of H by V_H , and the edge set of H by E_H , respectively. For non-empty $X \subset V$ and $F \subseteq E$, $\delta_F(X)$ denotes the set of hyperedges in F intersecting both X and $V \setminus X$. When $F = E$, we may represent $\delta_F(X)$ by $\delta(X)$. For some function $f : E \rightarrow \mathbb{Q}_+$ and $F \subseteq E$, $\sum_{e \in F} f(e)$ is represented by $f(F)$. For non-empty $X \subset V$, $E[X]$ denotes the set of

hyperedges in E contained in X , and $H[X]$ denotes the sub-hypergraph $(X, E[X])$ of H .

H is called ℓ -connected if $c(\delta(X)) \geq \ell$ for all non-empty $X \subset V$. H is called *connected* if $|\delta(X)| \geq 1$ for all non-empty $X \subset V$. Notice that the 1-connectedness is not equivalent to the connectedness.

A partition $\mathcal{V} = \{V_1, V_2, \dots, V_k\}$ of V into k non-empty subsets is called k -partition. We let $\delta(\mathcal{V}) = \cup_{i=1}^k \delta(V_i)$. H is called ℓ -partition-connected if $c(\delta(\mathcal{V})) \geq \ell(|\mathcal{V}| - 1)$ for all partitions $\mathcal{V}$ of V into non-empty subsets. It is easy to see that the ℓ -partition-connectivity is a stronger condition than the ℓ -connectivity. We call a partition $\mathcal{V}$ achieving $\min\{c(\delta(\mathcal{V})) / (|\mathcal{V}| - 1)\}$ *weakest*. $\min\{c(\delta(\mathcal{V})) / (|\mathcal{V}| - 1)\}$ is denoted by ℓ_H . If $|\delta(\mathcal{V})| \geq |\mathcal{V}| - 1$ for all $\mathcal{V}$, H is called *partition-connected*. Notice that the 1-partition-connectedness is not equivalent to the partition-connectedness.

If a k -cut of H is minimal, it is equal to $\delta(\mathcal{V})$ where $\mathcal{V}$ consists of the connected components after removing the k -cut. Hence the minimum multiway cut problem is equivalent to the problem of finding a k -partition $\mathcal{V}$ of H minimizing $c(\delta(\mathcal{V}))$. We call such a partition *minimum k -partition*.

A *hyperforest* in $H = (V, E)$ is defined as $F \subseteq E$ such that $|F[X]| \leq |X| - 1$ for every non-empty $X \subseteq V$. A hyperforest F is called *hypertree* if $|F| = |V| - 1$ and $\cup_{e \in F} e = V$. Notice that if $H = (V, E)$ is a graph, $F \subseteq E$ is a hypertree if and only if F is a spanning tree. Actually a hypertree is an extension of a spanning tree which inherits many important properties of spanning trees. However, there is also a difference between them. For example, in contrast to spanning trees, a connected hypergraph may contain no hypertree. Frank, Király and Kriesell [1] characterized hypergraphs containing hypertrees as follows.

Theorem 1 (Frank, Király, Kriesell [1]). *A hypergraph contains disjoint ℓ hypertrees if and only if it is ℓ -partition-connected.*

Let $\mathcal{F}$ be the family of hyperforests. The important building-block for proving Theorem 1 is the fact that $(E, \mathcal{F})$ is a matroid. Matroids defined from hypergraphs in such a way are called *hypergraphic matroids*. Hypertrees are bases of the hypergraphic matroid.

A hypertree packing of H is a pair of a set $\mathcal{T}$ of hypertrees in H and a non-negative weight $\alpha : \mathcal{T} \rightarrow \mathbb{Q}_+$ such that $\alpha(\mathcal{T}_e) \leq c(e)$ holds for all $e \in E$ where $\mathcal{T}_e$ denotes the set of hypertrees in $\mathcal{T}$ containing e . A hypergraph packing is called *maximum* if $\alpha(\mathcal{T})$ is maximum. We define the *packing value* of a hypergraph H as the maximum of $\alpha(\mathcal{T})$ over all hypertree packings of H .

3 Outline of our result

The first step of our result is to prove the following theorem originally proven for graphs by Thorup [12] while *recursively maximum hypertree packings* are defined in Section 4.

Theorem 2. *A recursively maximum hypertree packing of H contains a hypertree that shares at most $\gamma k - 3$ hyperedges with a minimum k -cut of H .*

Assume the hypertree and the $h = \gamma k - 3$ hyperedges in Theorem 2 are specified. Since each of the other $n - 1 - h$ hyperedges in the hypertree intersects only one elements of the minimum k -partition, shrinking them into single vertices preserves the minimum k -partition. If $\gamma = 2$, these $n - 1 - h$ hyperedges form at most $h + 1$ connected components. Hence the hypergraph obtained by shrinking them contains at most $h + 1$ vertices, for which the minimum k -partition can be found by enumerating all k -partitions.

If $\gamma \geq 3$, the number of the connected components formed by the $n - 1 - h$ hyperedges is not obvious because one large deleted hyperedge may connect many components. However, a characterization of hypertrees due to Lovász [8] tells that such a case does not occur even if $\gamma \geq 3$.

Theorem 3 (Lovász [8]). *Replace each hyperedge by an edge joining two vertices chosen from the hyperedge. It is possible to construct a spanning tree from an hypergraph by this operation if and only if the hypergraph is a hypertree.*

Corollary 1. *After removing h hyperedges from a hypertree, there exists at most $h + 1$ connected components.*

Proof. Consider the spanning tree constructed from a hypertree as shown by Theorem 3. After removing h edges from the tree, the remaining edges forms $h + 1$ connected components. The vertices in the same connected component are also connected by the hyperedges corresponding to the remaining edges. Hence removing h hyperedges from a hypertree results in at most $h + 1$ connected components. $\square$

Another point to note is the existence of hypertrees. As mentioned in Section 2, there exists connected hypergraphs which contain no hypertrees. For such hypergraphs, hypertree packings give no information on minimum k -cuts. Our algorithm avoids this situation by replacing each hyperedge $e \in E$ by its $|e|$ copies with capacity $c(e)/|e|$. Obviously this replacement makes no effect on minimum k -partitions while obtained hypergraphs contain hypertrees. Notice that after the replacement, the number of hyperedges are increased to at most γm .

Theorem 4. *Let $H' = (V, E')$ be the hypergraph obtained from a connected hypergraph $H = (V, E)$ by replacing each $e \in E$ by $|e|$ copies of e . Then H' contains a hypertree.*

Proof. Let $\mathcal{V}$ be a partition of V . Since each $e' \in \delta_{E'}(\mathcal{V})$ intersects at most $|e'|$ components of $\mathcal{V}$, $|\delta_{E'}(\mathcal{V})| \geq \sum_{U \in \mathcal{V}} \sum_{e' \in \delta_{E'}(U)} 1/|e'|$. Since each $e \in E$ has $|e|$ copies in E' , $\sum_{e' \in \delta_{E'}(U)} 1/|e'| = \sum_{e \in \delta_E(U)} 1 = |\delta_E(U)|$. Moreover, $|\delta_E(U)| \geq 1$ because H is connected. Thus $|\delta_{E'}(\mathcal{V})| \geq \sum_{U \in \mathcal{V}} 1 = |\mathcal{V}|$, which implies the partition-connectivity of H' . Hence by Theorem 1, H' contains a hypertree. $\square$

Now we describe our algorithm for the hypergraph multiway cut problem.

Algorithm 1: Minimum k -Cut Algorithm

Input: A connected hypergraph $H = (V, E)$ with capacity $c : E \rightarrow \mathbb{Q}_+$ and an integer $k \geq 3$

Output: A minimum k -cut of H

Step 1: For each $e \in E$, prepare $|e|$ copies $e_1, e_2, \dots, e_{|e|}$ of e with capacity $c(e_i) = c(e)/|e|$, $i \in \{1, 2, \dots, |e|\}$, and replace e by them.

Step 2: Define $F = E$. Compute a recursively maximum hypertree $(\mathcal{T}^*, \alpha^*)$ of H .

Step 3: For each $T \in \mathcal{T}^*$ and each set T' of $h = \gamma k - 3$ hyperedges in T , execute the following operations:

3-1: Compute a hypergraph H' obtained by shrinking all hyperedges in T' ;

3-2: Compute a minimum k -cut F' of H' ;

3-3: Let $F := F'$ if $c(F') \leq c(F)$.

Step 4: Output F .

For each hypertree in the recursively maximum hypertree packing, the algorithm runs in $O(k^{h+1}n^h)$ time. The recursively maximum hypertree packing is computable in strongly polynomial time, and the number of hypertrees in the packing is also strongly polynomial. Hence the strong polynomiality of the above algorithm can be proven.

To make the algorithm faster, Thorup's algorithm [12] constructs a set of spanning trees in a greedy way instead of computing a recursively maximum spanning tree packing. This technique also works in the hypergraphs obtained by replacing each hyperedge e with its $|e|$ copies. Let us first describe an algorithm for obtaining a set of hypertrees.

Algorithm 2: Greedy Algorithm for Computing a Set of Hypertrees

Input: A connected hypergraph $H = (V, E)$ with capacity $c : E \rightarrow \mathbb{Q}_+$ and an integer t .

Output: A set of t hypertrees of H .

Step 1: Let $\mathcal{T} := \emptyset$.

Step 2: Compute a minimum cost hypertree T of H with respects to the cost defined as $|\mathcal{T}_e|/c(e)$ for each $e \in E$, and $\mathcal{T} := \mathcal{T} \cup \{T\}$.

Step 3: If $|\mathcal{T}| = t$, then output $\mathcal{T}$. Otherwise, return to Step 2.

Theorem 5. *Let $H = (V, E)$ be a hypergraph such that each $e \in E$ has at least $|e| - 1$ copies in $E \setminus \{e\}$ of the same capacity. For this H , the greedy algorithm with $t = 24\gamma^4mk^3 \ln(2\gamma^2kmn)$ outputs a set of hypertrees which contains a hypertree sharing at most $h = \gamma k - 2$ hyperedges with a minimum k -cut of H .*

By Theorem 5, we can replace the computation of recursively maximum packings of hypertrees in Algorithm 1 by Algorithm 2 with $t = 24\gamma^4mk^3 \ln(2\gamma^2kmn)$. Moreover, choose sets of $\gamma k - 2$ hyperedges instead of sets of $\gamma k - 3$ hyperedges in Algorithm 1.

As mentioned in Section 2, hypertrees are bases of a hypergraphic matroid. Independence testing in a hypergraphic matroid can be done by minimizing a submodular function $f : 2^V \rightarrow \mathbb{Q}_+$ defined by $f(X) = |X| - |E[X]|$ for each $X \subseteq V$. It is possible to evaluate $f(X)$ in $O(\gamma m)$ for every $X \subseteq V$. Hence the time for computing a minimum cost hypertree is $O(\gamma m^2 \theta_n)$ where θ_n is the number of evaluating f for minimizing f . Thus by using the greedy algorithm instead of computing a recursively maximum hypertree packing, we have the following.

Corollary 2. *The hypergraph k -cut problem is solvable in $O(k^{\gamma k+2}n^{\gamma k-2}\gamma^5m^3\theta_n \log(k\gamma^2mn))$ time.*

In the subsequent sections, we prove Theorems 2 and 5.

4 Proof of Theorem 2

Let $\mathcal{V}$ be an arbitrary partition of V , and $(\mathcal{T}, \alpha)$ be an arbitrary hypertree packing of H . Since every hypertree $T \in \mathcal{T}$ satisfies $|T| = |V| - 1$ and contains at most $|U| - 1$ hyperedges contained by U for each $U \in \mathcal{V}$, it satisfies

$$|\delta_T(\mathcal{V})| = |T| - \sum_{U \in \mathcal{V}} |T[U]| \geq |V| - 1 - \sum_{U \in \mathcal{V}} (|U| - 1) = |\mathcal{V}| - 1. \quad (1)$$

Moreover,

$$c(e) \geq \alpha(\mathcal{T}_e) \quad \text{for each } e \in \delta(\mathcal{V}) \quad (2)$$

by the definition of hypertree packings. Thus it follows that

$$c(\delta(\mathcal{V})) \geq \sum_{e \in \delta(\mathcal{V})} \alpha(\mathcal{T}_e) = \sum_{T \in \mathcal{T}} \alpha(T) |\delta_T(\mathcal{V})| \geq (|\mathcal{V}| - 1) \sum_{T \in \mathcal{T}} \alpha(T). \quad (3)$$

Let $\mathcal{V}^*$ be a weakest partition of V , and $(\mathcal{T}^*, \alpha^*)$ be a maximum hypertree packing of H . Moreover, let M be a positive integer such that all of $Mc(e)$, $e \in E$ and $M\alpha^*(T)$, $T \in \mathcal{T}^*$ are integers. Notice that $(\mathcal{T}^*, M\alpha^*)$ is a maximum hypertree packing of the hypergraph H associated with hyperedge capacity Mc . Applying Theorem 1 to this hypergraph shows that $Mc(\delta(\mathcal{V}^*)) / (|\mathcal{V}^*| - 1) = \sum_{T \in \mathcal{T}^*} M\alpha^*(T)$ holds. That is to say, $\mathcal{V}^*$ and $(\mathcal{T}^*, \alpha^*)$ satisfy (3) with equality. We can derive the following lemma from this equality.

Lemma 1. $|\delta_T(\mathcal{V}^*)| = |\mathcal{V}^*| - 1$ holds for each $T \in \mathcal{T}^*$, and $\alpha^*(\mathcal{T}_e^*) = c(e)$ holds for each $e \in \delta(\mathcal{V}^*)$. Moreover, $T[U]$ is a hypertree on $H[U]$ for any pair of $T \in \mathcal{T}^*$ and $U \in \mathcal{V}^*$.

Proof. Since $\mathcal{V}^*$ and $(\mathcal{T}^*, \alpha^*)$ satisfy (3) with equality, they also satisfy (1) and (2), used for deriving (3), with equality. Let us prove that for any pair of $T \in \mathcal{T}^*$ and $U \in \mathcal{V}^*$, $T[U]$ is a hypertree on $H[U]$. Since T satisfies (1) with equality, T contains exactly $|U| - 1$ hyperedges in $E[U]$. For every $X \subseteq U$, $T[X]$ contains at most $|X| - 1$ hyperedges in $E[X]$ by the definition of T . For any $v \in U$, $|\{e \in T[U] \mid v \in e\}| = |T[U]| - |T[U \setminus \{v\}]| \geq (|U| - 1) - (|U| - 2) = 1$ holds. Hence $\cup_{e \in T[U]} e = U$. Therefore $T[U]$ is a hypertree on U . $\square$

Let $U \in \mathcal{V}^*$, $\mathcal{T}' = \{T[U] \mid T \in \mathcal{T}^*\}$, and α' be a weight defined on the hypertrees in $\mathcal{T}'$ such that $\alpha'(T[U]) = \alpha^*(T)$ for $T \in \mathcal{T}^*$. By Lemma 1, $(\mathcal{T}', \alpha')$ is a hypertree packing of $H[U]$. However, it may not be a maximum hypertree packing of $H[U]$ since the partition-connectivity of $H[U]$ may be larger than $\alpha'(\mathcal{T}')$. Let $(\mathcal{S}, \beta)$ be a maximum hypertree packing of $H[U]$. For each $T \in \mathcal{T}$ and $S \in \mathcal{S}$, replacing hyperedges in T contained by U with those in S generates another hypertree of H because hypertrees are bases of hypergraphic matroids. Hence from $(\mathcal{T}, \alpha)$ and $(\mathcal{S}, \beta)$, we can construct a maximum hypertree packing $(\mathcal{U}, \zeta)$ of H such that $|\mathcal{U}| \leq |\mathcal{T}| + |\mathcal{S}|$ and $(\mathcal{U}', \zeta')$ is a maximum hypertree packing of $H[U]$ where $\mathcal{U}' = \{T[U] \mid T \in \mathcal{U}\}$ and $\zeta'(T[U]) = \zeta(T)\beta(\mathcal{S})/\zeta(\mathcal{U})$ for each $T[U] \in \mathcal{U}'$. A maximum hypertree packing obtained by repeating this operation is called *recursively maximum*. That is to say, it can be obtained by the following algorithm.

Algorithm 3: Computing a Recursively Maximum Hypertree Packing

Input: A connected hypergraph $H = (V, E)$ with capacity $c : E \rightarrow \mathbb{Q}_+$.

Output: A recursively maximum hypertree packing of H .

Step 1: Compute a maximum hypertree packing $(\mathcal{T}^*, \alpha^*)$ of H , and a weakest partition $\mathcal{V}^*$ of H .

Step 2: While there exists $U \in \mathcal{V}^*$ such that $|U| > 1$, execute the following operations:

2-1: Compute a maximum hypertree packing $(\mathcal{S}, \beta)$ of $H[U]$, and a weakest partition $\mathcal{V}$ of $H[U]$;

2-2: For each $T \in \mathcal{T}^*$, execute the following operations: If there exists $S \in \mathcal{S}$ such that $\alpha^*(T) \leq \beta(S)$, then replace the hyperedges in $T[U]$ by those in S , and $\beta(S) := \beta(S) - \alpha^*(T)$; Otherwise, i.e., if there exists $S \in \mathcal{S}$ such that $\alpha^*(T) > \beta(S)$, then construct a hypertree $T' = (T \setminus T[U]) \cup S$ with $\alpha^*(T') := \beta(S)$, and update $\mathcal{T}^* \cup \{T'\}$, $\alpha^*(T) := \alpha^*(T) - \beta(S)$, and $\mathcal{S} := \mathcal{S} \setminus \{S\}$;

2-3: $\mathcal{V}^* := (\mathcal{V}^* \setminus \{U\}) \cup \mathcal{V}$.

Step 3: Output $(\mathcal{T}^*, \alpha^*)$.

From now on, we let $(\mathcal{T}^*, \alpha^*)$ stand for a recursively maximal hypertree packing. From $\mathcal{T}^*$ and given k , define $\mathcal{V}_k$ as the k -partition of V constructed by the following algorithm.

Algorithm 4: Computing $\mathcal{V}_k$

Input: A connected hypergraph $H = (V, E)$ with capacity $c : E \rightarrow \mathbb{Q}_+$, and an integer $k \geq 2$.

Output: A k -partition of V .

Step 1: Define $\mathcal{V}_k := \{V\}$.

Step 2: Let U be a set minimizing the partition-connectivity of $H[U]$ among those in $\mathcal{V}_k$ with $|U| \geq 2$. Compute a weakest partition $\mathcal{U} = \{U_1, U_2, \dots, U_{|\mathcal{U}|}\}$ of $H[U]$, where we assume that $\sum_{T \in \mathcal{T}^*} \alpha^*(T) |\delta(U_i) \cap T[U]| \leq \sum_{T \in \mathcal{T}^*} \alpha^*(T) |\delta(U_j) \cap T[U]|$ for $1 \leq i < j \leq |\mathcal{U}|$.

Step 3: If $|\mathcal{V}_k| - 1 + |\mathcal{U}| < k$, then $\mathcal{V}_k := (\mathcal{V}_k \setminus \{U\}) \cup \mathcal{U}$ and return to Step 2.

Step 4: If $|\mathcal{V}_k| - 1 + |\mathcal{U}| \geq k$, then $\mathcal{V}_k := (\mathcal{V}_k \setminus \{U\}) \cup \{U_1, U_2, \dots, U_{k-|\mathcal{V}_k|}, U \setminus \bigcup_{i=1}^{k-|\mathcal{V}_k|} U_i\}$, and output $\mathcal{V}_k$.

Lemma 2.

$$\sum_{e \in \delta(\mathcal{V}_k)} \alpha^*(\mathcal{T}_e^*) < (\gamma k - 2) \alpha^*(\mathcal{T}^*).$$

Proof. $\mathcal{V}'_k$ stands for $\mathcal{V}_k$ immediately before executing the operations Step 4 of Algorithm 4, and $\mathcal{V}_k$ stands for it outputted by Algorithm 4.

By the definition of $\mathcal{T}^*$ and Lemma 1, $T[U]$ is a hypertree of $H[U]$ for every pair of $U \in \mathcal{V}'_k$ and $T \in \mathcal{T}^*$. Thus $|\delta(\mathcal{V}'_k) \cap T| \leq |\mathcal{V}'_k| - 1$ holds for each $T \in \mathcal{T}^*$, and hence $\sum_{e \in \delta(\mathcal{V}'_k)} \alpha^*(\mathcal{T}_e^*) = \sum_{T \in \mathcal{T}^*} \alpha^*(T) |\delta(\mathcal{V}'_k) \cap T| \leq \alpha^*(\mathcal{T}^*) (|\mathcal{V}'_k| - 1)$ holds at this moment.

Let U be the element of $\mathcal{V}'_k$ and $\mathcal{U} = \{U_1, U_2, \dots, U_{|\mathcal{U}|}\}$ be the weakest partition of U computed in Step 2 immediately before executing Step 4. Note that $|\mathcal{U}| > k - |\mathcal{V}'_k|$ holds now by the condition of Step 4. By the same reason with above, $|\delta(\mathcal{U}) \cap T[U]| \leq |\mathcal{U}| - 1$ holds for each $T \in \mathcal{T}^*$. Hence $\sum_{T \in \mathcal{T}^*} \alpha^*(T) |\delta(\mathcal{U}) \cap T[U]| \leq (|\mathcal{U}| - 1) \alpha^*(\mathcal{T}^*)$.

Let $\mathcal{V}_U = \{U_1, U_2, \dots, U_{k-|\mathcal{V}'_k|}, U \setminus \bigcup_{j=1}^{k-|\mathcal{V}'_k|} U_j\}$. Then

$$\sum_{T \in \mathcal{T}^*} \alpha^*(T) |\delta(\mathcal{V}_U) \cap T[U]| \leq \sum_{j=1}^{k-|\mathcal{V}'_k|} \sum_{T \in \mathcal{T}^*} \alpha^*(T) |\delta(U_j) \cap T[U]|$$

Since the elements in $\mathcal{U}$ are sorted so that $\sum_{T \in \mathcal{T}^*} \alpha^*(T) |\delta(U_i) \cap T[U]| \leq \sum_{T \in \mathcal{T}^*} \alpha^*(T) |\delta(U_j) \cap T[U]|$ for $1 \leq i < j \leq |\mathcal{U}|$,

$$\sum_{j=1}^{k-|\mathcal{V}'_k|} \sum_{T \in \mathcal{T}^*} \alpha^*(T) |\delta(U_j) \cap T[U]| \leq \frac{k - |\mathcal{V}'_k|}{|\mathcal{U}|} \sum_{j=1}^{|\mathcal{U}|} \sum_{T \in \mathcal{T}^*} \alpha^*(T) |\delta(U_j) \cap T[U]|.$$

Since each hyperedge intersects at most γ elements in $\delta(\mathcal{U})$, it holds that

$$\begin{aligned} \frac{k - |\mathcal{V}'_k|}{|\mathcal{U}|} \sum_{j=1}^{|\mathcal{U}|} \sum_{T \in \mathcal{T}^*} \alpha^*(T) |\delta(U_j) \cap T[U]| &\leq \frac{k - |\mathcal{V}'_k|}{|\mathcal{U}|} \gamma \sum_{T \in \mathcal{T}^*} \alpha^*(T) |\delta(\mathcal{U}) \cap T[U]| \\ &\leq \frac{k - |\mathcal{V}'_k|}{|\mathcal{U}|} \gamma (|\mathcal{U}| - 1) \alpha^*(\mathcal{T}^*) \\ &\leq (k - |\mathcal{V}'_k|) \gamma \alpha^*(\mathcal{T}^*). \end{aligned}$$

Combining these implies that $\sum_{T \in \mathcal{T}^*} \alpha^*(T) |\delta(\mathcal{V}_U) \cap T[U]| \leq (k - |\mathcal{V}'_k|) \gamma \alpha^*(\mathcal{T}^*)$.

Notice that $\delta(\mathcal{V}_k) \cap T = (\delta(\mathcal{V}'_k) \cap T) \cup (\delta(\mathcal{V}_U) \cap T[U])$. Therefore it follows that

$$\sum_{e \in \delta(\mathcal{V}_k)} \alpha^*(\mathcal{T}_e^*) = \sum_{T \in \mathcal{T}^*} \alpha^*(T) |\delta(\mathcal{V}_k) \cap T| \leq \{(|\mathcal{V}'_k| - 1) + (k - |\mathcal{V}'_k|) \gamma\} \alpha^*(\mathcal{T}^*) < (\gamma k - 2) \alpha^*(\mathcal{T}^*).$$

□

Lemma 3. For each $e \in \delta(\mathcal{V}_k)$ and $f \in E \setminus \delta(\mathcal{V}_k)$,

$$\frac{\alpha^*(\mathcal{T}_e^*)}{c(e)} \geq \frac{\alpha^*(\mathcal{T}_f^*)}{c(f)}.$$

Proof. Let $U \in \mathcal{V}_k$ denote the set containing f (i.e., $f \in E[U]$). Let $\mathcal{V}'_k$ denote $\mathcal{V}_k$ immediately before e enters $\delta(\mathcal{V}_k)$ in Algorithm 4 for computing $\mathcal{V}_k$. Assume that e is contained by $U' \in \mathcal{V}'_k$ (i.e., $e \in E[U']$). Moreover, Let ℓ_U (resp., $\ell_{U'}$) denote the packing value of $H[U]$ (resp., $H[U']$).

From $(\mathcal{T}^*, \alpha^*)$, define $\mathcal{T}' = \{T[U'] \mid T \in \mathcal{T}^*\}$ and $\alpha'(T[U']) = \alpha^*(T) \ell_{U'} / \alpha^*(\mathcal{T}^*)$ for $T \in \mathcal{T}^*$. By the definition of recursively maximum hypertree packings, $(\mathcal{T}', \alpha')$ is a maximum hypertree packing of $H[U']$. By Lemma 1, the capacity constraint of edge e is tight for any maximum hypertree packing of $H[U']$, i.e., $c(e) = \alpha'(\mathcal{T}_e')$. Since $\alpha'(\mathcal{T}_e') = \alpha^*(\mathcal{T}_e^*) \ell_{U'} / \alpha^*(\mathcal{T}^*)$,

$$\ell_{U'} = c(e) \frac{\alpha^*(\mathcal{T}^*)}{\alpha^*(\mathcal{T}_e^*)}.$$

On the other hand, a maximum hypertree packing of $H[U]$ satisfies the capacity constraint for edge f . Hence, similarly with above,

$$\ell_U \leq c(f) \frac{\alpha^*(\mathcal{T}^*)}{\alpha^*(\mathcal{T}_f^*)}.$$

$\mathcal{V}'_k$ contains U'' such that $U \subseteq U''$. In other words, $U = U''$, or U is obtained by dividing U'' in Algorithm 4. As explained when recursively maximum hypertree packings are defined, $\ell_{U''} \leq \ell_U$ holds. Since Step 2 chose U' immediately before e enters $\delta(\mathcal{V}_k)$, $\ell_{U'} \leq \ell_{U''}$ holds. Therefore the required inequality has been proven. □

Let $\mathcal{V}^{\text{opt}}$ denote a minimum k -partition of H .

Lemma 4.

$$\sum_{T \in \mathcal{T}^*} \alpha^*(T) |\delta(\mathcal{V}^{\text{opt}}) \cap T| \leq \sum_{T \in \mathcal{T}^*} \alpha^*(T) |\delta(\mathcal{V}_k) \cap T|.$$

Proof. Define $\eta = \min_{e \in \delta(\mathcal{V}_k)} \alpha^*(\mathcal{T}_e^*) / c(e)$. By Lemma 3, each hyperedge $e \in \delta(\mathcal{V}^{\text{opt}}) \setminus \delta(\mathcal{V}_k)$ satisfies $\alpha^*(\mathcal{T}_e^*) / c(e) \leq \eta$. Hence it holds that

$$\sum_{e \in \delta(\mathcal{V}^{\text{opt}}) \setminus \delta(\mathcal{V}_k)} \alpha^*(\mathcal{T}_e^*) = \sum_{e \in \delta(\mathcal{V}^{\text{opt}}) \setminus \delta(\mathcal{V}_k)} c(e) \frac{\alpha^*(\mathcal{T}_e^*)}{c(e)} \leq \eta c(\delta(\mathcal{V}^{\text{opt}}) \setminus \delta(\mathcal{V}_k)).$$

The definition of $\mathcal{V}^{\text{opt}}$ implies that $c(\delta(\mathcal{V}^{\text{opt}})) \leq c(\delta(\mathcal{V}_k))$, and hence $c(\delta(\mathcal{V}^{\text{opt}}) \setminus \delta(\mathcal{V}_k)) \leq c(\delta(\mathcal{V}_k) \setminus \delta(\mathcal{V}^{\text{opt}}))$. Thus

$$\begin{aligned}
\sum_{T \in \mathcal{T}^*} \alpha^*(T) |\delta(\mathcal{V}^{\text{opt}}) \cap T| &= \sum_{e \in \delta(\mathcal{V}^{\text{opt}}) \cap \delta(\mathcal{V}_k)} \alpha^*(\mathcal{T}_e^*) + \sum_{e \in \delta(\mathcal{V}^{\text{opt}}) \setminus \delta(\mathcal{V}_k)} \alpha^*(\mathcal{T}_e^*) \\
&\leq \sum_{e \in \delta(\mathcal{V}^{\text{opt}}) \cap \delta(\mathcal{V}_k)} \alpha^*(\mathcal{T}_e^*) + \eta c(\delta(\mathcal{V}^{\text{opt}}) \setminus \delta(\mathcal{V}_k)) \\
&\leq \sum_{e \in \delta(\mathcal{V}^{\text{opt}}) \cap \delta(\mathcal{V}_k)} \alpha^*(\mathcal{T}_e^*) + \eta c(\delta(\mathcal{V}_k) \setminus \delta(\mathcal{V}^{\text{opt}})) \\
&\leq \sum_{e \in \delta(\mathcal{V}_k)} \alpha^*(\mathcal{T}_e^*) \\
&= \sum_{T \in \mathcal{T}^*} \alpha^*(T) |\delta(\mathcal{V}_k) \cap T|.
\end{aligned}$$

□

From Lemmas 2 and 4, we can observe that

$$\frac{\sum_{T \in \mathcal{T}^*} \alpha^*(T) |\delta(\mathcal{V}^{\text{opt}}) \cap T|}{\alpha^*(\mathcal{T}^*)} < \gamma k - 2.$$

This means that some hypertree T in $\mathcal{T}^*$ satisfies

$$|\delta(\mathcal{V}^{\text{opt}}) \cap T| < \gamma k - 2.$$

Therefore Theorem 2 has been proven.

5 Proof of Theorem 5

In this section, we present a proof of Theorem 5. Although it is almost same with that for $\gamma = 2$ presented by Thorup [12], we describe all details for self-containment.

Let $H = (V_H, E_H)$ be a hypergraph such that each $e \in E_H$ has at least $|e| - 1$ copies in $E_H \setminus \{e\}$ of the same capacity. Note that $|E_H| = \gamma m$. We denote the capacity of hyperedges in H by c_H in order to avoid misunderstanding. Moreover, we assume that a recursively maximum hypertree packing $(\mathcal{T}^*, \alpha^*)$ of H satisfies $\alpha^*(\mathcal{T}_e^*) = \alpha^*(\mathcal{T}_{e'}^*)$ for $e \in E_H$ and its copy $e' \in E_H$.

For a set $\mathcal{T}$ of hypertrees of H and $e \in E_H$, we let $u_H^{\mathcal{T}}(e)$ denote $|\mathcal{T}| / (c_H(e) |\mathcal{T}|)$. For each $e \in E$, we also define $u_H^*(e)$ as $\alpha^*(\mathcal{T}_e^*) / (c_H(e) \alpha^*(\mathcal{T}^*))$ from a recursively maximum hypertree packing $(\mathcal{T}^*, \alpha^*)$ of H . Since $c_H(e) \geq \alpha^*(\mathcal{T}_e^*)$ for all $e \in E_H$, $1/u_H^*(e)$ is at least the packing value of H , i.e., $1/u_H^*(e) \geq \alpha^*(\mathcal{T}^*)$. Moreover, since $c_H(e) = \alpha^*(\mathcal{T}_e^*)$ holds for some $e \in E_H$ by the maximality of $(\mathcal{T}^*, \alpha^*)$, $\min_{e \in E_H} 1/u_H^*(e) = \alpha^*(\mathcal{T}^*)$ holds.

Recall that Algorithm 3 updates $\mathcal{V}^*$ by partitioning non-singleton sets in $\mathcal{V}^*$ repeatedly until no such set exists. For $e \in E_H$, define U_e as the last set in $\mathcal{V}^*$ such that $e \in E_H[U_e]$ during the execution of the algorithm. Then $\max_{e' \in E_H[U_e]} u_{H[U_e]}^*(e') = u_{H[U_e]}^*(e)$. The definition of recursively maximum hypertree packings implies that $u_{H[U_e]}^*(e') = u_H^*(e')$ for each $e' \in E_H[U_e]$ because $\alpha^*(\mathcal{T}_{e'}^*) / \alpha^*(\mathcal{T}^*) = \beta(\mathcal{T}_{e'}) / \beta(\mathcal{T})$ holds with a maximum hypertree packing $(\mathcal{T}, \beta)$ of $H[U_e]$. Therefore, the partition-connectivity of $H[U_e]$ is $1/u_H^*(e)$.

Lemma 5. *Let I be a subgraph of H and assume that each hyperedge e in I has a capacity $c_I(e)$ such that $c_{\min} \leq c_I(e) \leq c_H(e)$. Let $C = \sum_{e \in E_I} c_I(e)$, and $u_I = \max_{e \in E_I} u_I^*(e)$. Moreover, let*

ϵ be an arbitrary real such that $0 < \epsilon < 1/2$, and $\mathcal{T}^g$ be a set of hypertrees in H constructed by Algorithm 2 with $t \geq 3 \ln(C/c_{\min})/(c_{\min}u_I\epsilon^2)$. Then

$$u_H^{\mathcal{T}^g}(e) < (1 + \epsilon)u_I \quad (4)$$

holds for each $e \in E_I$.

Proof. Scaling hyperedge capacities makes no effect on the claim. Hence we assume without loss of generality that $c_{\min} = 1$.

Let $\mathcal{T}$ denote a set of hypertrees kept by Algorithm 2 at some moment during it is running for computing $\mathcal{T}^g$. The key is the following quantity:

$$\sum_{e \in E_I} c_I(e) \frac{(1 + \epsilon)^{|\mathcal{T}_e|/c_H(e)} (1 + \epsilon u_I)^{t - |\mathcal{T}|}}{(1 + \epsilon)^{(1 + \epsilon)u_I t}}. \quad (5)$$

We prove the following three facts about this quantity:

- (i) When $\mathcal{T} = \emptyset$, (5) is less than 1;
- (ii) If (5) is less than 1 when $|\mathcal{T}| = t$, then (4) holds for all $e \in E_I$;
- (iii) When a tree is added to $\mathcal{T}$ in Step 2 of Algorithm 2, then (5) is not increased.

Clearly these three facts imply (4) for all $e \in E_I$.

Proof of (i): If $\mathcal{T} = \emptyset$, then (5) is equal to

$$\sum_{e \in E_I} c_I(e) \frac{(1 + \epsilon u_I)^t}{(1 + \epsilon)^{(1 + \epsilon)u_I t}} < C \frac{e^{\epsilon u_I t}}{(1 + \epsilon)^{(1 + \epsilon)u_I t}} < C e^{-\epsilon^2 u_I t/3} \leq 1.$$

Notice that the last inequality holds by $t \geq 3 \ln(C/c_{\min})/(c_{\min}u_I\epsilon^2)$.

Proof of (ii): When $|\mathcal{T}| = t$, (5) is equal to

$$\sum_{e \in E_I} c_I(e) (1 + \epsilon)^{|\mathcal{T}_e|/c_H(e) - (1 + \epsilon)u_I t}.$$

Remember that $c_I(e) \geq c_{\min} = 1$ for each $e \in E_I$. Hence (5) < 1 means that $|\mathcal{T}_e|/c_H(e) - (1 + \epsilon)u_I t < 0$ for every $e \in E_I$. Thus

$$u_H^{\mathcal{T}^g}(e) = \frac{|\mathcal{T}_e|}{t \cdot c_H(e)} < (1 + \epsilon)u_I$$

holds for every $e \in E_I$.

Proof of (iii): Let T denote the tree added to $\mathcal{T}$ in Step 2 of Algorithm 2, and $\mathcal{T}' = \mathcal{T} \cup \{T\}$. We define

$$q(T) = \sum_{e \in E_I} c_I(e) (1 + \epsilon)^{|\mathcal{T}'_e|/c_H(e)}.$$

For proving (iii), it suffices to show that $q(T) \leq \sum_{e \in E_I} c_I(e) (1 + \epsilon)^{|\mathcal{T}_e|} (1 + \epsilon u_I)$.

Define

$$q'(T) = \sum_{e \in E_I} c_I(e) (1 + \epsilon)^{|\mathcal{T}_e|/c_H(e)} \left(1 + \frac{\epsilon \cdot \Delta(e, T)}{c_I(e)} \right)$$

where $\Delta(e, T) = 1$ if $e \in T$, and $\Delta(e, T) = 0$ otherwise. Elementary calculations imply that $q(T) \leq q'(T)$. Hence we prove that $q'(T) \leq \sum_{e \in E_I} c_I(e)(1 + \epsilon)^{|\mathcal{T}_e|} (1 + \epsilon u_I)$.

Let $(\mathcal{S}^*, \beta^*)$ be a recursively maximum hypertree packing of I . Then

$$\begin{aligned} \frac{\sum_{T' \in \mathcal{S}^*} \beta^*(T') q'(T')}{\beta^*(\mathcal{S}^*)} &= \sum_{e \in E_I} c_I(e)(1 + \epsilon)^{|\mathcal{T}_e|/c_H(e)} \left(1 + \frac{\epsilon}{c_I(e)} \frac{\beta^*(\mathcal{S}_e^*)}{\beta^*(\mathcal{S}^*)} \right) \\ &\leq \sum_{e \in E_I} c_I(e)(1 + \epsilon)^{|\mathcal{T}_e|/c_H(e)} (1 + \epsilon u_I). \end{aligned}$$

Thus $\mathcal{S}^*$ contains a hypertree T' of I such that $q'(T') \leq \sum_{e \in E_I} c_I(e)(1 + \epsilon)^{|\mathcal{T}_e|/c_H(e)} (1 + \epsilon u_I)$.

Let us observe that T satisfies $q'(T) \leq q'(T')$ for every $T' \in \mathcal{S}^*$. $q'(T) \leq q'(T')$ is equivalent to $c'(T \cap E_I) \leq c'(T')$ where the capacities c' of the hyperedges is defined by $c'(e) = (1 + \epsilon)^{|\mathcal{T}_e|/c_H(e)}$ for each $e \in E_I$. Remember that T is a minimum hypertree of H with regards to the hyperedge cost defined by $|\mathcal{T}_e|/c_H(e)$ for $e \in E_H$. If e and e' in E_H satisfies $|\mathcal{T}_e|/c_H(e) \leq |\mathcal{T}_{e'}|/c_H(e')$, then $c'(e) \leq c'(e')$ holds.

Since hypertrees are bases of the hypergraphic matroid, a minimum hypertree can be obtained by choosing hyperedges greedily in the non-decreasing order of the costs. Hence $T \cap E_I$ is a subset of a minimum hypertree T'' of I , and $c'(T'') \leq c'(T')$. This proves the lemma. $\square$

From Lemma 5, we have the following fact.

Lemma 6. *Let $0 < \epsilon \leq 1/2$, and $\mathcal{T}^g$ be a set of hypertrees of H constructed by Algorithm 2 with $|\mathcal{T}^g| = t \geq 3\gamma m \ln(\gamma mn/\epsilon)/\epsilon^3$. Then*

$$|\mathcal{T}_e^g| \leq \frac{1 + \epsilon}{1 - \epsilon} \cdot \frac{\alpha^*(\mathcal{T}_e^*)}{\alpha^*(\mathcal{T}^*)} \cdot |\mathcal{T}^g| + 1$$

holds for each $e \in E$.

Proof. Choose an arbitrary hyperedge $\tilde{e} \in E_H$. We prove the required inequality for this hyperedge.

Now we construct a subgraph of H , to which we apply Lemma 5. Let $I_0 = H[U_{\tilde{e}}]$. We define the capacity $c_0(e)$ of $e \in E_{I_0}$ in I_0 as $c_H(e)$, and denote $\max_{e \in E_{I_0}} u_{I_0}^*(e)$ by u_0 . Note that $u_{I_0}^*(e) = u_H^*(e)$ for each $e \in E_{I_0}$, and $u_0 = u_H^*(\tilde{e})$.

Define $c_{\min} = \epsilon/(\gamma m u_0)$. Next, construct I_1 from I_0 by removing all hyperedges of capacity less than $c_{\min}$. The capacities c_1 of the remaining hyperedges in I_1 are not changed from c_0 . Note that $\tilde{e}$ may not be contained by I_1 . Since removing hyperedges does not increase the partition-connectivity of the hypergraph, $1/u_1 \leq 1/u_0$ holds where u_1 is defined as $\max_{e \in E_{I_1}} u_{I_1}^*(e)$. Observe that the decrease of the partition connectivity is at most $\gamma m c_{\min} = \epsilon/u_0$. This implies that $1/u_1 \geq (1 - \epsilon)/u_0$.

In the last, construct I_2 from I_1 by reducing the capacities of hyperedges greedily as long as possible while keeping $u_2 = u_1$ and $c_2(e) \geq c_{\min}$ for every $e \in E_{I_2}$ where $u_2 = \max_{e \in E_{I_2}} u_{I_2}^*(e)$ and c_2 is the obtained hyperedge capacity in I_2 .

When the reduction stops, each $e \in E_{I_2}$ satisfies $u_{I_2}^*(e) = u_2 (= u_1)$. For proving this fact, assume that there are some hyperedges e such that $u_{I_2}^*(e) < u_1$. Let I' be a connected component of the subgraph of I_2 consisting of such hyperedges. The partition-connectivity of I' is larger than $1/u_1$, i.e., $\min_{e \in E_{I'}} 1/u_{I'}^*(e) > 1/u_1$. $E_{I'}$ contains a hyperedge e' such that $c_2(e') > c_{\min}$ because otherwise, the partition-connectivity of I' is at most $\gamma m c_{\min} = \epsilon/u_0 \leq (1 - \epsilon)/u_0 \leq 1/u_1$,

a contradiction. Reducing the capacity of such e' keeps $u_2 = u_1$ since e' is contained by no multiway cut defined from a weakest partition of I_2 . This is also a contradiction to the fact that the reduction stops.

Now we obtain a subgraph I_2 of H , which satisfies $u_2 = u_{I_2}^*(e)$ for each $e \in E_{I_2}$, $c_2(e) \geq c_{\min}$ for each $e \in E_{I_2}$, and $u_H^*(\tilde{e}) \leq u_2 \leq u_H^*(\tilde{e})/(1 - \epsilon)$. Moreover, $\sum_{e \in E_{I_2}} c_2(e) = (|V_{I_2}| - 1)/u_2 < n/u_0 = \gamma n m c_{\min}/\epsilon$. Let $\mathcal{T}^g$ be a set of t hypertrees of H constructed by the greedy algorithm where t is defined so that it satisfies

$$\frac{3 \ln(C/c_{\min})}{c_{\min} u_2 \epsilon^2} \leq \frac{3 \ln(\gamma n m / \epsilon)}{c_{\min} u_2 \epsilon^2} \leq \frac{3 \gamma m \ln(\gamma n m / \epsilon)}{\epsilon^3} \leq t.$$

Applying Lemma 5 to I_2 , we can see that each $e \in E_{I_2}$ satisfies $u_H^{\mathcal{T}^g}(e) \leq (1 + \epsilon)u_{I_2}(e) \leq \frac{1 + \epsilon}{1 - \epsilon} u_H^*(\tilde{e})$. Hence if $\tilde{e} \in E(I_2)$, then we have

$$|\mathcal{T}_e^g| = u_H^{\mathcal{T}^g}(\tilde{e}) c_H(\tilde{e}) |\mathcal{T}^g| \leq \frac{1 + \epsilon}{1 - \epsilon} u_H^*(\tilde{e}) c_H(\tilde{e}) |\mathcal{T}^g| = \frac{1 + \epsilon}{1 - \epsilon} \cdot \frac{\alpha^*(\mathcal{T}_e^*)}{\alpha^*(\mathcal{T}^*)} \cdot |\mathcal{T}^g|.$$

Consider the case where $\tilde{e} \notin E_{I_2}$. Let T' be the hypertree which is added to $\mathcal{T}^g$ most lately among those containing $\tilde{e}$. Let $\mathcal{T}'$ be the set of hypertrees added to $\mathcal{T}^g$ before T' . By the above discussion,

$$|\mathcal{T}'_e| \leq |\mathcal{T}_e^g| \leq \frac{1 + \epsilon}{1 - \epsilon} u_H^*(e) c_H(e) |\mathcal{T}^g|$$

holds for each $e \in E_{I_2}$. Recall that T' is a minimum hypertree with respect to the cost defined by $|\mathcal{T}'_e|/c_H(e)$ for each $e \in E_H$. Since $\tilde{e}$ is contained by T' , the cost of $\tilde{e}$ is at most the cost of e if $\tilde{e}$ is in the fundamental circuit of e in the hypergraphic matroid defined from H .

For any $\emptyset \neq X \subset V_{I_2}$, $\delta_{E_{I_2}}(X) \neq \emptyset$ because

$$c_{I_1}(\delta_{E_{I_2}}(X)) \geq c_H(\delta_{E_H}(X)) - \gamma m c_{\min} \geq (1 - \epsilon)/u_0 > 0$$

and $\delta_{E_{I_2}}(X) = \delta_{E_{I_1}}(X)$. This means that I_2 is connected. Moreover, if $e \in E_{I_2}$, then all copies e' of e in H is also contained by E_{I_2} since $c_H = (e) = c_H(e')$ and $\alpha_H^*(e) = \alpha_H^*(e')$. Hence I_2 is partition-connected. By Theorem 1, I_2 contains at least one hypertree. Moreover, $\tilde{e} \subseteq V_{I_2}$. Thus, even if $\tilde{e}$ is not contained by E_{I_2} , there exists a hyperedge $e \in E_{I_2}$ such that $|\mathcal{T}'_e|/c_H(\tilde{e}) \leq |\mathcal{T}'_e|/c_H(e)$ since otherwise, $\tilde{e}$ is not contained by T' . Therefore, with such e , $\tilde{e}$ satisfies

$$\frac{|\mathcal{T}'_{\tilde{e}}|}{c_H(\tilde{e})} \leq \frac{|\mathcal{T}'_e|}{c_H(e)} \leq |\mathcal{T}^g| \cdot u_H^*(e) \cdot \frac{1 + \epsilon}{1 - \epsilon}.$$

Since $|\mathcal{T}_e^g| = |\mathcal{T}'_e| + 1$, we have

$$|\mathcal{T}_e^g| \leq |\mathcal{T}^g| \cdot u_H^*(e) \cdot c_H(\tilde{e}) \cdot \frac{1 + \epsilon}{1 - \epsilon} + 1 \leq |\mathcal{T}^g| \cdot u_H^*(\tilde{e}) \cdot c_H(\tilde{e}) \cdot \frac{1 + \epsilon}{1 - \epsilon} + 1 = |\mathcal{T}^g| \cdot \frac{\alpha^*(\mathcal{T}_e^*)}{\alpha^*(\mathcal{T}^*)} \cdot \frac{1 + \epsilon}{1 - \epsilon} + 1.$$

□

Lemma 6 proves Theorem 5 as follows. Let $\mathcal{V}^{\text{opt}}$ stand for the minimum k -partition of H . Lemma 6 shows that

$$\sum_{T \in \mathcal{T}^g} |\delta(\mathcal{V}^{\text{opt}}) \cap T| = \sum_{e \in \delta(\mathcal{V}^{\text{opt}})} |\mathcal{T}_e^g| \leq \sum_{e \in \delta(\mathcal{V}^{\text{opt}})} \left(t \frac{1 + \epsilon}{1 - \epsilon} \cdot \frac{\alpha^*(\mathcal{T}_e^*)}{\alpha^*(\mathcal{T}^*)} + 1 \right) \leq t \frac{1 + \epsilon}{1 - \epsilon} \cdot \frac{\sum_{e \in \delta(\mathcal{V}^{\text{opt}})} \alpha^*(\mathcal{T}_e^*)}{\alpha^*(\mathcal{T}^*)} + \gamma m.$$

In the last of Section 4, we have observed that

$$\frac{\sum_{e \in \delta(\mathcal{V}_k)} \alpha^*(\mathcal{T}_e^*)}{\alpha^*(\mathcal{T}^*)} = \frac{\sum_{T \in \mathcal{T}^*} \alpha^*(T) |\delta(\mathcal{V}^{\text{opt}}) \cap T|}{\alpha^*(\mathcal{T}^*)} < \gamma k - 2.$$

These mean that

$$\frac{\sum_{T \in \mathcal{T}^g} |\delta(\mathcal{V}^{\text{opt}}) \cap T|}{t} < \frac{1+\epsilon}{1-\epsilon}(\gamma k - 2) + \frac{\gamma m}{t}.$$

Recall that $t = 3\gamma m \ln(\gamma mn/\epsilon)/\epsilon^3$. Assume that $n, m \geq 2$. Then $t \geq 6\gamma m/\epsilon^3$, and hence the right-hand side of the above inequality is at most

$$\frac{1+\epsilon}{1-\epsilon}(\gamma k - 2) + \epsilon^3/6 = \gamma k - 2 + 2\gamma k\epsilon + \epsilon \left(\frac{2k\gamma\epsilon - 4}{1-\epsilon} + \frac{\epsilon^2}{6} \right).$$

Setting ϵ to $1/(4k)$, the right-hand side is at most $\gamma k - 1$, which means that

$$\frac{\sum_{T \in \mathcal{T}^g} |\delta(\mathcal{V}^{\text{opt}}) \cap T|}{t} < \gamma k - 1.$$

This implies that $\mathcal{T}^g$ contains a hypertree T such that $|\delta(\mathcal{V}^{\text{opt}}) \cap T| < \gamma k - 1$. Moreover, $t = 3\gamma m \ln(\gamma mn/\epsilon)/\epsilon^3 = 24\gamma^4 m k^3 \ln(2\gamma^2 k m n)$. Therefore the proof is completed.

6 Discussion

Our algorithm proposed in this paper is not polynomial if γ is not fixed. A reason for this fact is that Theorem 2 depends on γ . If we can remove γ from Theorem 2, we have a polynomial algorithm even if γ is not fixed. However the following example shows that it is impossible.

Define a set V of vertices as $\{v_1, v_2, \dots, v_n\}$. We sometimes represent v_i by v_{i+n} with $i \in \{1, 2, \dots, n\}$, for convenience. We also define a set E of hyperedges as $\{e_1, e_2, \dots, e_{n-1}\}$ where a hyperedge e_i is defined as $\{v_i, v_{i+1}, \dots, v_{i+\gamma}\}$. Let us consider the hypergraph $G = (V, E)$ with uniform hyperedge capacity.

G is a hypertree. Hence the set of hypertrees in a recursively maximum hypertree packing of G consists of only E . On the other hand, a minimum k -cut of G is $[\{v_n\}, \{v_1\}, \dots, \{v_{k-2}\}, \{v_{k-1}, \dots, v_{n-1}\}]$. These hypertree and k -cut share $\gamma + k - 3$ hyperedges.

There is still a gap between $\gamma + k - 3$ and $\gamma k - 3$. Nevertheless, this example tells that recursively maximum hypertree packings do not give an algorithm whose computational time is independent from γ .

Acknowledgements

This work was partially supported by Grant-in-Aid for Scientific Research from the Ministry of Education, Culture, Sports, Science and Technology of Japan.

References

- [1] A. Frank, T. Király, and M. Kriesell. On decomposing a hypergraph into k connected sub-hypergraphs. *Discrete Applied Mathematics*, 131(2):373–383, 2003.
- [2] N. Garg, V. V. Vazirani, and M. Yannakakis. Multiway cuts in node weighted graphs. *Journal of Algorithms*, 50:49–61, 2004.
- [3] L. Gasieniec, J. Jansson, A. Lingas, and A. Óstlin. On the complexity of constructing evolutionary trees. *Journal of Combinatorial Optimization*, 3:183–197, 1999.

- [4] A. V. Goldberg and R. E. Tarjan. A new approach to the maximum flow problem. *Journal of the ACM*, 35:921–940, 1988.
- [5] O. Goldschmidt and D. Hochbaum. A polynomial algorithm for the k -cut problem for fixed k . *Mathematics of Operations Research*, 19:24–37, 1994.
- [6] Y. Kamidoi, N. Yoshida, and H. Nagamochi. A deterministic algorithm for finding all minimum k -way cuts. *SIAM Journal on Computing*, 36:1329–1341, 2006.
- [7] D. R. Karger and C. Stein. A new approach to the minimum cut problem. *Journal of the ACM*, 43:601–640, 1996.
- [8] L. Lovász. A generalization of könig’s theorem. *Acta. Math. Acad. Sci. Hungar.*, 21:443–446, 1970.
- [9] H. Nagamochi. Algorithms for the minimum partitioning problems in graphs. *IEICE Transactions on Information and Systems*, J86-D-1:53–68, 2003.
- [10] K. Okumoto, T. Fukunaga, and H. Nagamochi. Divide-and-conquer algorithms for partitioning hypergraphs and submodular systems. Technical report, Department of Applied Mathematics and Physics, Graduate School of Informatics, Kyoto University, 2009.
- [11] M. Thorup. Fully-dynamic min-cut. *Combinatorica*, 27:91–127, 2007.
- [12] M. Thorup. Minimum k -way cuts via deterministic greedy tree packing. In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing*, pages 159–166, 2008.
- [13] M. Xiao. Finding minimum 3-way cuts in hypergraphs. In *Theory and Applications of Models of Computation: Proceedings of the 5th International Conference*, Lecture Notes in Computer Science 4978, pages 270–281, 2008.
- [14] M. Xiao. An improved divide-and-conquer algorithm for finding all minimum k -way cuts. In *Algorithms and Computation: Proceedings of the 19th International Symposium*, Lecture Notes in Computer Science 5369, pages 208–219, 2008.
- [15] L. Zhao, H. Nagamochi, and T. Ibaraki. A unified framework for approximating multiway partition problems. In *Proceedings of the 12th International Symposium on Algorithms and Computation*, Lecture Notes in Computer Science 2223, pages 682–694, 2001.