An Improved Divide-and-Conquer Algorithm for Finding All Minimum k-Way Cuts*

Mingyu Xiao

School of Computer Science and Engineering
University of Electronic Science and Technology of China
Chengdu 610054, China
myxiao@gmail.com

Abstract. Given a positive integer k and an edge-weighted undirected graph $G = (V, E; w)$, the *minimum k-way cut* problem is to find a subset of edges of minimum total weight whose removal separates the graph into k connected components. This problem is a natural generalization of the classical *minimum cut* problem and has been well-studied in the literature.

A simple and natural method to solve the minimum k-way cut problem is the divide-and-conquer method: getting a minimum k-way cut by properly separating the graph into two small graphs and then finding minimum k'-way cut and k''-way cut respectively in the two small graphs, where $k' + k'' = k$. In this paper, we present the first algorithm for the tight case of $k' = \lfloor k/2 \rfloor$. Our algorithm runs in $O(n^{4k-\lg k})$ time and can enumerate all minimum k-way cuts, which improves all the previously known divide-and-conquer algorithms for this problem.

Keywords: k-Way Cut, Divide-and-Conquer, Graph Algorithm.

1 Introduction

Let k be a positive integer and $G = (V, E; w)$ a connected undirected graph where each edge e has a positive weight $w(e)$. A *k-way cut* of G is a subset of edges whose removal separates the graph into k connected components, and the *minimum k-way cut* problem is to find a k-way cut of minimum total weight. The minimum k-way cut problem is a natural generalization of the classical *minimum cut* problem and has great applications in the area of VLSI system design, parallel computing systems, clustering, network reliability and finding cutting planes for the travelling salesman problems.

The minimum 2-way cut problem is commonly known as the *minimum cut* problem and can be solved in $O(mn+n^2 \log n)$ time by Nagamochi and Ibaraki's algorithm [12] or Stoer and Wagner's algorithm [19]. Another version of the minimum 2-way cut problem is the *minimum (s, t) cut* problem, which asks us to

* The work was done when the author was a PhD student in Department of Computer Science and Engineering, the Chinese University of Hong Kong.

S.-H. Hong, H. Nagamochi, and T. Fukunaga (Eds.): ISAAC 2008, LNCS 5369, pp. 208–219, 2008.

find a minimum cut that separates two given vertices s and t. The minimum (s,t) cut problem can be solved in $O(mn\log n^2/m)$ time by Goldberg and Tarjan's algorithm [4] and $O(\min(n^{2/3},m^{1/2})m\log(n^2/m)\log U)$ time by Goldberg and Rao's algorithm [3], where U is the maximum capacity of the edge. Finding a minimum cut or minimum (s,t) cut is a subroutine in our algorithms. In the remainder of the paper, we use $T(n,m)=\widetilde{O}(mn)$ to denote the running time of computing a minimum cut or a minimum (s,t) cut in an edge-weighted graph.

For $k=3$, Kamidoi et al. [8] and Kapoor [10] showed that the minimum 3-way cut problem can be solved by computing $O(n^3)$ minimum (s,t) cuts. Later, Burlet and Goldschmidt [1] improved this result to $O(n^2)$ minimum cut computations. He [6] showed that in unweighted planar graphs the minimum 3-way cut problem can be solved in $O(n\log n)$ time. Xiao [22] designed the first polynomial algorithm for finding minimum 3-way cuts in hypergraphs.

Furthermore, Kamidoi et al. [8] and Nagamochi and Ibaraki [13] proved that the minimum 4-way cut problem can be solved by computing $O(n)$ minimum 3-way cuts. Nagamochi et al. [14] extended this result for minimum $\{5,6\}$-way cuts by showing that $T_k(n,m)=O(nT_{k-1}(n,m))$, where $k=5,6$, and $T_k(n,m)$ is the running time of computing a minimum k-way cut. Those results lead to $\widetilde{O}(mn^k)$ time algorithms for the minimum k-way cut problem for $k\leq 6$.

For general k, Goldschmidt and Hochbaum [5] proved that the minimum k-way cut problem is NP-hard when k is part of the input and gave the first polynomial algorithm for fixed k. The running time of their algorithm is $O(n^{k^2}T(n,m))$. Later, Kamidoi et al. [9] improved the running time to $O(n^{4k/(1-1.71/\sqrt{k})-34}T(n,m))$. Karger and Stein [11] proposed a Monte Carlo algorithm with $O(n^{2(k-1)}\log^3 n)$ running time. Recently, Thorup [20] designed an deterministic algorithm with running time $O(n^{2k})$, which is based on tree packing. Since this problem is NP-hard for arbitrary k, it is also interesting to design approximation algorithms for it. Saran and Vazirani[18] gave two simple approximation algorithms with ratio of $(2-2/k)$ and running time of $O(nT(n,m))$. Naor and Rabani [16] obtained an integer program formulation of this problem with integrality gap 2, and Ravi and Sinha [17] also derived a 2-approximation algorithm via the network strength method. Zhao et al. [24] proved that the approximation ratio is $2-3/k$ for an odd k and $2-(3k-4)/(k^2-k)$ for an even k, if we compute a k-way cut of the graph by iteratively finding and deleting minimum 3-way cuts in the graph. Xiao et al. [23] determined the tight approximation ratio of a general greedy splitting algorithm, in which we iteratively increase a constant number of components of the graph with minimum cost. That result implies that the approximation ratio is $2-h/k+O(h^2/k^2)$ for the algorithm that iteratively increases $h-1$ components.

Most deterministic algorithms for finding minimum k-way cuts, including the two algorithms presented by Goldschmidt and Hochbaum [5] and Kamidoi et al. [9], are based on a divide-and-conquer method. The main idea is to get a minimum k-way cut by properly separating the graph into two small graphs and then finding minimum k'-way cut and k''-way cut respectively in the two small graphs, where $k'+k''=k$. We say that cut $C=[X,\overline{X}]$ is an $(h,k-h)$-*cut* of

G, if there is a minimum k-way cut $C_k = [Y_1, \cdots, Y_h, Y_{h+1} \cdots, Y_k]$ of G such that $\bigcup_{i=1}^{h} Y_i = X$ and $\bigcup_{i=h+1}^{k} Y_i = \overline{X}$. Once an $(h, k-h)$-cut $C = [X, \overline{X}]$ is given, we only need to find a minimum h-way cut in induced subgraph $G[X]$ and a minimum $(k-h)$-way cut in induced subgraph $G[\overline{X}]$. Goldschmidt and Hochbaum [5] proved that there are a set S of at most $k-2$ vertices and a set T of at most $k-1$ vertices such that a minimum (S,T) cut is a $(1, k-1)$-cut. By enumerating all the possibilities of S and T, we have at most $O(n^{2k-3})$ candidates for $(1, k-1)$-cuts. Goldschmidt and Hochbaum obtained an $O(n^{k^2})$ algorithm for the minimum k-way cut problem by recursively applying this method. There are two ways to improve this method. First, we can reduce the sizes of S and T. Second, we can try to make minimum (S,T) cut a more 'balanced' cut, in other words, we want minimum (S,T) cut an $(h, k-h)$-cut such that h is close to $\frac{k}{2}$. Kamidoi et al. [9] proved that there are a set S of at most $k-2$ vertices and a set T of at most $k-2$ vertices such that a minimum (S,T) cut is a $(p, k-p)$-cut with $p = \lceil (k - \sqrt{k})/2 \rceil - 1$, and then they got an $O(n^{4k/(1-1.71/\sqrt{k})-34} T(n,m))$ algorithm for the minimum k-way cut problem. In this paper, we show that there are a set S of at most $2\left\lfloor \frac{k}{2} \right\rfloor$ vertices and a set T of at most $k-1$ vertices such that a minimum (S,T) cut is a $(\left\lfloor \frac{k}{2} \right\rfloor, \left\lceil \frac{k}{2} \right\rceil)$-cut. Based on this property, we obtain an $O(n^{4k-\lg k})$ algorithm for finding all minimum k-way cuts. Previous results as well as our result are summarized in the following table. Recently Thorup [20] designed an even faster algorithm for the minimum k-way cut problem, which is based on tree packing, but not the divide-and-conquer method.

Table 1. History of divide-and-conquer algorithms for the minimum k-way cut problem

	Goldschmidt et al. [5]	Kamidoi et al. [9]	This paper
Bounds on $\lvert S\rvert$ and $\lvert T\rvert$	$k-2$ and $k-1$	$k-2$ and $k-2$	$2\lfloor k/2 \rfloor$ and $k-1$
The min (S,T) cut	$(1, k-1)$-cut	$(p, k-p)$-cut, $p = \lceil (k-\sqrt{k})/2 \rceil - 1$	$(\lfloor k/2 \rfloor, \lceil k/2 \rceil)$-cut
Running time for the min k-way cut problem	$O(n^{k^2})$	$O(n^{4k/(1-1.71/\sqrt{k})-16})$	$O(n^{4k-\lg k})$

In this paper, we assume the original graph $G = (V, E; w)$ is a connected graph with more than k vertices. For an edge subset $E' \subseteq E$, $w(E')$ denotes the total weight of the edges in E'. Let $X_1, X_2, \cdots, X_l \subset V$ be l $(2 \leq l \leq n)$ disjoint nonempty subsets of vertices, then $[X_1, X_2, \cdots, X_l]$ denotes the set of edges crossing any two different vertex sets of $\{X_1, X_2, \cdots, X_l\}$. A 2-way cut is also simply called a *cut* of the graph. Cut $[X, \overline{X}]$ is called an (S,T) *cut*, if $S \subseteq X$ and $T \subseteq \overline{X}$. Sometimes a singleton set $\{s\}$ is simply written as s and $w([X_1, X_2, \cdots, X_l])$ as $w(X_1, X_2, \cdots, X_l)$. The rest of the paper is organized as follows: We first present the simple divide-and-conquer algorithm in Section 2. Then we give the proofs of our structural results in Section 3. In the last section, we conclude with some remarks.

2 The Divide-and-Conquer Algorithm

Let $C = [X, \overline{X}]$ be a cut. Recall that cut C is an $(h, k-h)$-cut of G if there is a minimum k-way cut $C_k = [Y_1, \cdots, Y_h, Y_{h+1} \cdots, Y_k]$ of G such that $X = \sum_{i=1}^{h} Y_i$ and $\overline{X} = \sum_{i=h+1}^{k} Y_i$. Let $C_k = [Y_1, \cdots, Y_k]$ be a minimum k-way cut and $1 \leq h \leq k-1$ an integer. By arbitrarily choosing h components $\{Y_{j_1}, Y_{j_2}, \cdots, Y_{j_h}\}$ of C_k, we get an $(h, k-h)$-cut $[\bigcup_{i=1}^{h} Y_{j_i}, \overline{\bigcup_{i=1}^{h} Y_{j_i}}]$, which is called an $(h, k-h)$-*partition* of C_k. Among all $(h, k-h)$-partitions, those with minimum weight are called *minimum* $(h, k-h)$*-partitions* of C_k and the weight of them is denoted by $w_{h,k-h}(C_k)$.

For an $(h, k-h)$-cut $[X, \overline{X}]$ of graph G, a minimum h-way cut $[Y_1, \cdots, Y_h]$ in induced graph $G[X]$ and a minimum $(k-h)$-way cut $[Z_1, \cdots, Z_{k-h}]$ in induced graph $G[\overline{X}]$ together yields a minimum k-way cut $[Y_1, \cdots, Y_h, Z_1, \cdots, Z_{k-h}]$ in the original graph G. This suggests a recursive way to solve the minimum k-way cut problem: find an $(h, k-h)$-cut $[X, \overline{X}]$ and recursively find minimum h-way and $(k-h)$-way cuts respectively in $G[X]$ and $G[\overline{X}]$.

However it is not easy to find an $(h, k-h)$-cut, even for $h = 1$. In Section 3, we will prove that for each minimum $(\lfloor \frac{k}{2} \rfloor, \lceil \frac{k}{2} \rceil)$-partition $[X, \overline{X}]$ of each minimum k-way cut, there are a set S of at most $2\lfloor \frac{k}{2} \rfloor$ vertices and a set T of at most $k-1$ vertices such that a minimum (S, T) cut is $[X, \overline{X}]$ (See Theorem 2). This minimum (S, T) cut is called the *nearest minimum* (S, T) *cut of* S and can be found by using the same time of computing a maximum flow from S to T. Theorem 2 enables us to obtain the following divide-and-conquer algorithm to find minimum k-way cuts. We enumerate all possibilities of S and T and find the nearest minimum (S, T) cuts in the graph. Then we get a family Γ of at most $\binom{n}{2\lfloor \frac{k}{2} \rfloor} \times \binom{n}{k-1} < n^{2\lfloor \frac{k}{2} \rfloor + (k-1)}$ (S, T) cuts by using the same number of maximum flow computations. By Theorem 2, Γ contains all minimum $(\lfloor \frac{k}{2} \rfloor, \lceil \frac{k}{2} \rceil)$-partitions of all minimum k-way cuts. We then recursively find, for each member of Γ, minimum $\lfloor \frac{k}{2} \rfloor$-way cut in $G[X]$ and minimum $\lceil \frac{k}{2} \rceil$-way cut in $G[\overline{X}]$. The minimum ones among all k-way cuts we find will be returned as our solution. The algorithm is described in Figure 1.

The correctness of algorithm **Multiwaycut** follows from Theorem 2. Now we analyze the running time. When $k = 2$, we use Nagamochi et al.'s algorithm [15] to find all minimum cuts directly, which runs in $O(m^2 n + mn^2 \log n) = O(mT(n, m))$ time. When $k > 2$, we get the recurrence relation

$$C(n, k) \leq n^{2\lfloor \frac{k}{2} \rfloor} + k - 1 (C(n, \lfloor \tfrac{k}{2} \rfloor) + C(n, \lceil \tfrac{k}{2} \rceil)) + n^{2\lfloor \frac{k}{2} \rfloor} + k - 1, \qquad (1)$$

where $C(n, k)$ is the upper bound on the number of maximum flow computations to be computed when algorithm **Multiwaycut** runs on an n-vertex graph and an integer k. It is easy to verify that $C(n, k) = O(n^{4k - \lg k - 3})$ satisfies (1) by using the substitution method.

Theorem 1. *All minimum k-way cuts can be found in $O(n^{4k - \lg k})$ time.*

Multiwaycut(G, k)

Input: A graph $G = (V, E; w)$ with nonnegative edge weights and a positive integer $k \leq |V|$.
Output: The set R of all minimum k-way cuts and the weight W of the minimum k-way cut.

1. If $\{k = 2\}$, **then** return the set of all the minimum cuts and the weight directly.
2. Else $\{k \geq 3\}$,
Let W be $+\infty$.
For each pair of disjoint nonempty vertex subsets S and T with $|S| \leq 2\left\lfloor \frac{k}{2} \right\rfloor$ and $|T| \leq k-1$, **do**
Compute the nearest minimum (S,T) cut $C = [X, \overline{X}]$ of S.
If $\{|X| \geq \left\lfloor \frac{k}{2} \right\rfloor$ and $|\overline{X}| \geq \left\lceil \frac{k}{2} \right\rceil\}$, **then**
$(R_1, W_1) \longleftarrow$ **Multiwaycut**$(G[X], \left\lfloor \frac{k}{2} \right\rfloor)$.
$(R_2, W_2) \longleftarrow$ **Multiwaycut**$(G[\overline{X}], \left\lceil \frac{k}{2} \right\rceil)$.
If $\{W > w(C) + W_1 + W_2\}$, **then**
$W \longleftarrow w(C) + W_1 + W_2$,
$R \longleftarrow \{C \bigcup F_1 \bigcup F_2 \mid F_1 \in R_1, F_2 \in R_2\}$.
Else if $\{W = w(C) + W_1 + W_2\}$, **then**
$R \longleftarrow R \bigcup \{C \bigcup F_1 \bigcup F_2 \mid F_1 \in R_1, F_2 \in R_2\}$.
Return (R, W).

Fig. 1. The Algorithm **Multiwaycut**(G, k)

3 Structural Properties

In this section, we prove the following key theorem, which is the foundation of our divide-and-conquer algorithm.

Theorem 2. *Let C_k be a minimum k-way ($k \geq 3$) cut of a graph G and $[A, B]$ a minimum $(\left\lfloor \frac{k}{2} \right\rfloor, \left\lceil \frac{k}{2} \right\rceil)$-partition of C_k. Then there exits a set $S \subseteq A$ of at most $2\left\lfloor \frac{k}{2} \right\rfloor$ vertices and a set $T \subseteq B$ of at most $k-1$ vertices such that the nearest minimum (S,T) cut of S is $[A, B]$.*

To prove this theorem, we will derive some useful structural properties. Given two disjoint vertex sets S and T, a minimum (S,T) cut separates the graph into two components. One that contains S is called the *source part* and the other one is called the *sink part*, which contains T. For most cases, there are more than one minimum (S,T) cut. Among all minimum (S,T) cuts, the unique one that makes the source part of the maximum cardinality is called the *farthest minimum* (S,T) *cut* of S, and the unique one that makes the sink part of the maximum cardinality is called the *nearest minimum* (S,T) *cut* of S. The farthest minimum (S,T) cut of S is the same as the nearest minimum (T,S) cut of T. Ford and Fullkerson [7] proved the uniqueness of the farthest and nearest minimum (S,T) cuts by using the Max flow/Min cut theorem. We can easily get the farthest

and nearest minimum (S,T) cuts in linear time based on a maximum flow from S to T. (Note that given a maximum flow, in the residual graph, let X be the set of vertices who are connected with t. Then $[V-X,X]$ is the farthest minimum isolating cut for s). These two special minimum (S,T) cuts have been discussed and used in the literature [7], [21], [5], [2]. Next, we give more structural properties of them.

Lemma 1. *Let* $[X_1,\overline{X_1}]$ *be the nearest minimum* (S_1,T) *cut of* S_1 *and* $[X_2,\overline{X_2}]$ *the nearest minimum* (S_2,T) *cut of* S_2*, if* $S_1 \supseteq S_2$*, then* $X_1 \supseteq X_2$*.*

Lemma 2. *Let* $[X_1,\overline{X_1}]$ *be the farthest minimum* (S,T_1) *cut of* S *and* $[X_2,\overline{X_2}]$ *the farthest minimum cut* (S,T_2) *of* S*, if* $T_1 \supseteq T_2$*, then* $X_1 \subseteq X_2$*.*

Lemma 1 and Lemma 2 can be proved easily by using the uniqueness of the nearest and farthest minimum (s,t) cuts.

Lemma 3. *Let* $C_1 = [X_1,\overline{X_1}]$ *be the nearest minimum* (S_1,T) *cut of* S_1 *and* $C_2 = [X_2,\overline{X_2}]$ *a minimum* (S_2,T) *cut. If* $S_1 \subseteq X_2$*, then* $X_1 \subseteq X_2$*.*

Proof. Let $Z = X_1 - X_2$, $Y = X_2 - X_1$, $U = X_1 \cap X_2$, and $W = \overline{X_1 \cup X_2}$ (See Figure 2). To prove $X_1 \subseteq X_2$, we only need to prove that $Z = \emptyset$. Assume to the contrary that $Z \neq \emptyset$. We show the contradiction that $[X_1 - Z, \overline{X_1} + Z]$ is a 'nearer' minimum (S_1,T) cut of S_1 than $C_1 = [X_1,\overline{X_1}]$. Obviously, we only need to prove that $w(X_1 - Z, \overline{X_1} + Z) \leq w(S_1,T)$.

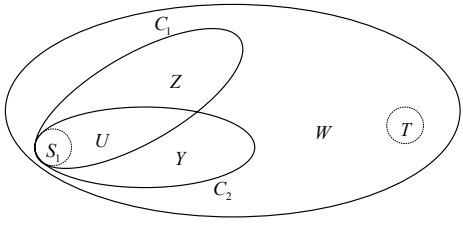

Fig. 2. Illustration for the proof of Lemma 3

Since $[U+Y+Z,W]$ is an (S_2,T) cut and $C_2 = [X_2,\overline{X_2}]$ a minimum (S_2,T) cut, we have

$$w(U+Y+Z,W) \geq w(X_2,\overline{X_2}).$$

It is clear that

$$[U+Y+Z,W] = [U+Y,W] + [Z,W]$$

and

$$[X_2,\overline{X_2}] = [U+Y,W+Z] = [U+Y,W] + [U,Z] + [Y,Z].$$

We get

$$w(U,Z) + w(Y,Z) \leq w(Z,W).$$

Therefore, $w(U, Y+W+Z) = w(U, Y+W)+w(U, Z) \leq w(U, Y+W)+w(Z, W) \leq w(U, Y+W)+w(Z, Y+W) = w(U+Z, Y+W) = w(C_1)$.

We will use the following relation between two multi-way cuts, which was proved by Xiao et al. in [23].

Proposition 1. *Given an edge-weighted graph G, and integers h and k ($2 \leq h \leq k$), then for any minimum h-way cut C_h and any k-way cut C_k of G, the following relation holds*

$$w(C_h) \leq \frac{(2k-h)(h-1)}{k(k-1)} w(C_k). \tag{2}$$

Kamidoi et al. [9] proved the following two important results

Proposition 2. *Given an edge-weighted graph G and two integers h and k ($1 \leq h < k$), let C_k be a minimum k-way cut in G and $w_{h,k-h}(C_k)$ the weight of the minimum $(h, k-h)$-partitions of C_k, then*

$$w_{h,k-h}(C_k) \leq \frac{2h(k-h)}{k(k-1)} w(C_k). \tag{3}$$

Proposition 3. *Given a graph $G = (V, E)$ with at least 4 vertices, two disjoint nonempty subsets T and R of V, and an integer $p \geq 2$, let $\{s_1, s_2, \cdots, s_p\} = S \subseteq V - T \cup R$ be a set of p vertices such that, for each $i \in \{1, 2, \cdots, p\}$, there is a minimum $(S \cup R - \{s_i\}, T)$ cut $[X_i, \overline{X_i}]$ which satisfies $(T \cup \{s_i\}) \subseteq \overline{X_i}$. Let $Z = \bigcap_{1 \leq i \leq p} X_i$, $W = \bigcup_{1 \leq i < j \leq p} (\overline{X_i} \cap \overline{X_j})$, and $Y_i = \overline{X_i} - W$ $(i = 1, 2, \cdots, p)$, then $C^\star = [Z, Y_1, Y_2, \cdots, Y_p, W]$ is a $(p+2)$-way cut such that*

$$w(C^\star) + w(Z, W) + w(Y_1, Y_2, \cdots, Y_p) \leq w(X_1, \overline{X_1}) + w(X_2, \overline{X_2}). \tag{4}$$

Based on Proposition 3, we can prove the following Lemma 4. The detailed proof can be found in the full version of this paper.

Lemma 4. *Given a graph $G = (V, E)$ with at least 4 vertices, a nonempty subset of vertices $T \subset V$, and an integer $p \geq 2$, let $\{s_1, s_2, \cdots, s_p\} = S \subseteq V - T$ be a set of p vertices such that, for each $i \in \{1, 2, \cdots, p\}$, there is a minimum $(S - \{s_i\}, T)$ cut $[X_i, \overline{X_i}]$ which satisfies $(T \cup \{s_i\}) \subseteq \overline{X_i}$. Let $Z = \bigcap_{1 \leq i \leq p} X_i$, $W = \bigcup_{1 \leq i < j \leq p} (\overline{X_i} \cap \overline{X_j})$, and $Y_i = \overline{X_i} - W$ $(i = 1, 2, \cdots, p)$,*

(a) : When $Z \neq \emptyset$, then $C^\star = [Z, Y_1, Y_2, \cdots, Y_p, W]$ is a $(p+2)$-way cut such that

$$w(C^\star) + w(Z, W) + w(Y_1, Y_2, \cdots, Y_p) \leq 2w(V - T, T). \tag{5}$$

(b) : When $Z = \emptyset$ and $p \geq 3$, then $C^\star = [Y_1, Y_2, \cdots, Y_p, W]$ is a $(p+1)$-way cut such that

$$w(C^\star) + \frac{p-3}{p+1} \cdot w(Y_1, Y_2, \cdots, Y_p) \leq \frac{p}{p+1} \cdot 2w(V - T, T). \tag{6}$$

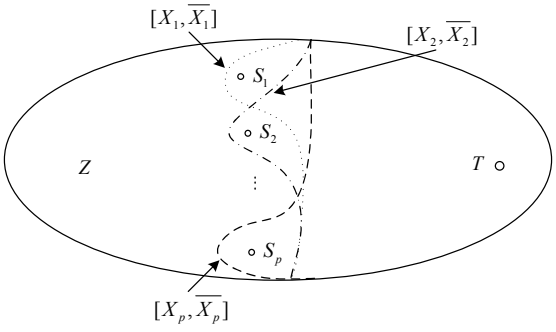

Fig. 3. Illustration for Proposition 3 and Lemma 4

Lemma 5. *Given a graph G and an integer $k \geq 3$, let w_k be the weight of a minimum k-way cut of G. For any cut $[A, B]$ in G with weight $w(A, B) < \frac{k}{2(k-1)} w_k$ (respectively, $w(A, B) < \frac{k+1}{2k-1} w_k$), there exists a set $S \subseteq A$ of at most $k-1$ (respectively, k) vertices such that the nearest minimum (S, B) cut of S is $[A, B]$.*

Proof. Let $k' = k - 1$ or k (for the two cases respectively), we only need to consider the case that $|A| \geq k' + 1$ (when $|A| < k' + 1$, we can just let $S = A$). Our proof includes two phases. In the first phase, we prove that if the lemma does not hold, then we can find a set $S_0 \subseteq A$ of $k'+1$ vertices such that, for each nonempty subset S'_0 of S_0, the nearest minimum (S'_0, B) cut $C' = [Z, \overline{Z}]$ of S'_0 satisfying $S'_0 \subseteq Z$ and $(S_0 - S'_0 + B) \subseteq \overline{Z}$. Then S_0 is a vertex set that satisfies the conditions in Lemma 4. In the second phase, based on S_0, we will show that there is a k-way cut with weight less than w_k, which is a contradiction.

Phase 1: finding S_0. We will give a procedure to select some vertices from A into S. Initially, all the vertices in A are unmarked and S is an empty set. Once a vertex is selected into S, we mark it. Sometimes a vertex in S_0 will also be removed from S, but this vertex is still remained as marked. First, we arbitrarily select $k' + 1$ vertices in A into S. For each nonempty subset $S' \subset S$, we check the nearest minimum (S', B) cut $C' = [Z, \overline{Z}]$ of S'. If $(S - S' + B) \not\subset \overline{Z}$, say $a \in (S - S') \bigcap Z$, we update S by removing a from S and adding an unmarked vertex into S (When there are no more unmarked vertex in $V - S$, we just remove a from S and stop the procedure). Once S is updated, we check all nonempty subsets S''s of S again. Since A is a finite set and in each iteration, one more vertex is marked, we will find a set S of $k'+1$ vertices that satisfies the conditions in Lemma 4 or no more unmarked vertex can be added into S. For the former case, we just let $S_0 = S$. For the later case, we show that S is a set of k' vertices such that the nearest minimum (S, B) cut of S is $[A, B]$, and thus the lemma holds. Let $S^{(1)}, S^{(2)}, \cdots, S^{(l_0)}$ be the updated sequence of S. Let $C^{(l)} = [Z_l, \overline{Z_l}]$ and $C^{(l+1)} = [Z_{l+1}, \overline{Z_{l+1}}]$ be the nearest minimum $(S^{(l)}, B)$ and $(S^{(l+1)}, B)$ cuts of $S^{(l)}$ and $S^{(l+1)}$ respectively. Since $S^{(l)} \subset Z_{l+1}$, we know that $Z_l \subseteq Z_{l+1}$ by

Lemma 3. For each $1 \leq l \leq l_0 - 1$, we have $Z_l \subseteq Z_{l+1}$. Then all the marked vertices will be in Z_{l_0}, where $[Z_{l_0}, \overline{Z_{l_0}}] = [A, B]$ is the nearest minimum $(S^{(l_0)}, B)$ cut of $S^{(l_0)}$. Furthermore, since $S^{(l_0)}$ is obtained by removing a vertex a from $S^{(l_0-1)}$, we know that the size of $S^{(l_0)}$ is k'.

Phase 2: finding a k-way cut with weight less than w_k based on S_0. Suppose $S_0 = \{s_1, \cdots, s_{k'}, s_{k'+1}\}$. Let $[X_i, \overline{X_i}]$ be the nearest minimum $(S - \{s_i\}, B)$ cut $(i = 1, \cdots, k'+1)$, then $[X_i, \overline{X_i}]$ satisfies that $(B \cup \{s_i\}) \subseteq \overline{X_i}$. Let $Z = \bigcap_{1 \leq i \leq k'+1} X_i$, $W = \bigcup_{1 \leq i < j \leq k'+1} (\overline{X_i} \cap \overline{X_j})$, and $Y_i = \overline{X_i} - W$ $(i = 1, \cdots, k'+1)$. Next, we consider $Z = \emptyset$ and $Z \neq \emptyset$ such two cases.

When $Z = \emptyset$, it follows from Lemma 4 that $C^\star = [Y_1, Y_2, \cdots, Y_{k'+1}, W]$ is a $(k'+2)$-way cut such that

$$w(C^\star) + \frac{k'-2}{k'+2} \cdot w(Y_1 + \cdots + Y_{k'+1}) \leq \frac{k'+1}{k'+2} \cdot 2w(A, B). \tag{7}$$

Since $C^\star$ is a $(k'+2)$-way cut and $k'+2 > k$, by Proposition 1, we know that there is a k-way cut C_k with weight

$$w(C_k) \leq \frac{(2(k'+2)-k)(k-1)}{(k'+2)(k'+1)} w(C^\star). \tag{8}$$

It follows from (7) and (8) that

$$w(C_k) \leq \frac{(2(k'+2)-k)(k-1)}{(k'+2)(k'+1)} \cdot \frac{k'+1}{k'+2} 2w(A, B).$$

In the case of $w(A, B) < \frac{k}{2(k-1)} w_k$, we have $k' = k-1$. Then

$$w(C_k) \leq \frac{(k+2)(k-1)}{(k+1)k} \cdot \frac{k}{k+1} \cdot 2w(A, B) < \frac{(k+2)k}{(k+1)^2} w_k < w_k.$$

In the case of $w(A, B) < \frac{k+1}{2k-1} w_k$, we have $k' = k$. Then

$$w(C_k) \leq \frac{(k+4)(k-1)}{(k+2)(k+1)} \cdot \frac{k+1}{k+2} \cdot 2w(A, B) < \frac{2(k+4)(k^2-1)}{(k+2)^2(2k-1)} w_k < w_k.$$

We get a contraction that C_k is k-way cut with weight less than w_k.

When $Z \neq \emptyset$, it follows from Lemma 4 that $C^\star = [Z, Y_1, Y_2, \cdots, Y_{k'+1}, W]$ is a $(k'+3)$-way cut such that

$$w(C^\star) + w(Z, W) + w(Y_1, \cdots, Y_{k'+1}) \leq 2w(A, B). \tag{9}$$

Suppose $w(Y_{i_0}, W) \geq w(Y_{i_1}, W) \geq w(Y_{i_2}, W) \geq \max_{i \neq i_0, i_1, i_2} \{w(Y_i)\}$. For the case of $w(A, B) < \frac{k}{2(k-1)} w_k$, we prove that $C_k = [Z, \mathcal{Y}_{-i_0-i_1}, W + Y_{i_0} + Y_{i_1}]$ is a k-way cut with weight less than w_k, where $\mathcal{Y}_{-i_0-i_1} = \{Y_1, \cdots, Y_{i_0-1}, Y_{i_0+1}, \cdots, Y_{i_1-1}, Y_{i_1+1}, \cdots, Y_{k'+1}\}$. For the case of $w(A, B) < \frac{k+1}{2k-1} w_k$, we prove that $C_k = [Z, \mathcal{Y}_{-i_0-i_1-i_2}, W + Y_{i_0} + Y_{i_1} + Y_{i_2}]$ is a k-way cut with weight less than w_k, where $\mathcal{Y}_{-i_0-i_1-i_2}$ is defined by the same way as $\mathcal{Y}_{-i_0-i_1}$.

In the case of $w(A,B) < \frac{k}{2(k-1)}w_k$, we have $k' = k-1$. Then $C_k = [Z, \mathcal{Y}_{-i_0-i_1}, W + Y_{i_0} + Y_{i_1}]$ is a k-way cut. Since $[X_i, \overline{X_i}]$ is a minimum $(S - \{s_i\}, B)$ cut, we have $w(Z, Y_i) \leq w(Y_i, W)$ for each $i \in \{1, 2, \cdots, k\}$. Therefore,

$$\begin{aligned} w(Y_{i_0} + Y_{i_1}, W) &\geq \frac{2}{2k}(\sum\nolimits_{i=1}^{k} w(Y_i, W) + \sum\nolimits_{i=1}^{k} w(Z, Y_i)) \\ &= \frac{1}{k}(w(C^\star) - w(Z, W) - w(Y_1, \cdots, Y_k)). \end{aligned}$$

By using this inequality and (9), we get

$$\begin{aligned} w(C_k) &\leq w(C^\star) - w(Y_{i_0} + Y_{i_1}, W) \\ &\leq \frac{k-1}{k}w(C^\star) + \frac{1}{k}w(Z, W) + \frac{1}{k}w(Y_1, \cdots, Y_k) \\ &\leq \frac{k-1}{k}2w(A, B) < \frac{k-1}{k}\frac{k}{k-1}w_k = w_k. \end{aligned}$$

In the case of $w(A,B) < \frac{k+1}{2k-1}w_k$, we have $k' = k$. Clearly, $C_k = [Z, \mathcal{Y}_{-i_0-i_1-i_2}, W + Y_{i_0} + Y_{i_1} + Y_{i_2}]$ is still a k-way cut. We get

$$\begin{aligned} w(Y_{i_0} + Y_{i_1} + Y_{i_2}, W) &\geq \frac{3}{2(k+1)}(\sum\nolimits_{i=1}^{k+1} w(Y_i, W) + \sum\nolimits_{i=1}^{k+1} w(Z, Y_i)) \\ &= \frac{3}{2(k+1)}(w(C^\star) - w(Z, W) - w(Y_1, \cdots, Y_{k+1})). \end{aligned}$$

By using this inequality and (9), we get

$$\begin{aligned} w(C_k) &\leq w(C^\star) - w(Y_{i_0} + Y_{i_1} + Y_{i_2}, W) \\ &\leq \frac{2k-1}{2(k+1)}w(C^\star) + \frac{3}{2(k+1)}w(Z, W) + \frac{3}{2(k+1)}w(Y_1, \cdots, Y_{k+1}) \\ &\leq \frac{2k-1}{2(k+1)}2w(A, B) < \frac{2k-1}{2(k+1)}\frac{2(k+1)}{2k-1}w_k = w_k. \end{aligned}$$

We have proved that, in both cases, there is a k-way cut with weight less than w_k. Thus, we have finished the proof.

Lemma 6. *Given a graph G and an integer $k \geq 3$, let w_k be the weight of the minimum k-way cut of G, then for any cut $[A, B]$ in G with weight $w(A,B) \leq \frac{k}{2(k-1)}w_k$ (respectively, $w(A,B) \leq \frac{k+1}{2k-1}w_k$), there exists a set $S \subseteq A$ of at most $k-1$ (respectively, k) vertices such that the farthest minimum (S, B) cut of S is $[A, B]$.*

In Lemma 6, the weight of the cuts can equal $\frac{k}{2(k-1)}w_k$ or $\frac{k+1}{2k-1}w_k$ and there are the farthest minimum (S, B) cuts, instead of the nearest minimum (S, B) cuts in Lemma 5. The proof of Lemma 6 just follows the proof of Lemma 5. Note that since in Lemma 6 there are farthest minimum (S, B) cuts, we have $w(X_i, \overline{X_i}) < w(V - T, T)$ for each $i \in \{1, 2, \cdots, p\}$. Therefore, The equal signs in (5) and (6) will not hold, which guarantees that the remaining part of the proof of Lemma 5 is suitable for Lemma 6.

Now we are ready to prove Theorem 2:

Proof. By Proposition 2, we have $w(A,B) \leq \frac{2h(k-h)}{k(k-1)} w_k$, where $h = \lfloor \frac{k}{2} \rfloor$. Since $\lfloor \frac{k}{2} \rfloor \lceil \frac{k}{2} \rceil \leq (\frac{k}{2})^2$, we have $w(A,B) \leq \frac{k}{2(k-1)} w_k$ and the equal sign does not hold for odd k. By Lemma 5 and Lemma 6, we know that there is a set $S \subseteq A$ of at most $2\lfloor \frac{k}{2} \rfloor$ vertices such that the nearest minimum (S,B) cut of S is $[A,B]$ and there is a set $T \subseteq B$ of at most $k-1$ vertices such that the farthest minimum (T,A) cut of T is $[B,A]$. We look at the nearest minimum (S,T) cut $[X,\overline{X}]$ of S. Since $[A,B]$ is the nearest minimum (S,B) cut of S and $T \subseteq B$, we have $A \subseteq X$ by Lemma 1. Since $[B,A]$ is the farthest minimum (T,A) cut of T and $S \subseteq A$, we have $B \subseteq \overline{X}$ by Lemma 2. Therefore, $[X,\overline{X}] = [A,B]$. Thus, Theorem 2 holds.

Corollary 1. *Given a graph G and an integer $k \geq 3$, let w_k be the weight of the minimum k-way cut of G, then the number of cuts with weight less than $\frac{k}{2(k-1)} w_k$ in G is bounded by n^{2k-2} and the number of cuts with weight less than $\frac{k+1}{2k-1} w_k$ is bounded by n^{2k}.*

4 Discussion

In this paper, we presented a simple divide-and-conquer algorithm for finding minimum k-way cuts. As we mentioned in Section 1, there are two possible ways to improve the algorithm. One is to reduce the sizes of S and T, and the other one is to make the minimum (S,T) cut be a more 'balanced' $(h, k-h)$-cut. In our algorithm, $h = \lfloor \frac{k}{2} \rfloor$ and this means our $(h, k-h)$-cuts are the most 'balanced'. Our questions are: For the most 'balanced' case, can we reduce the sizes of S and T? What are the lower bounds on the two sizes? Note that if we can reduce the two sizes to $\frac{k}{2}$, then the divide-and-conquer algorithm will run in $O(n^{2k})$ time.

Nagamochi et al. [13], [14] proved that the minimum k-way cut problem can be solved by computing $O(n)$ minimum $(k-1)$-way cuts for $k \leq 6$. Does this hold for general k? If so, then the minimum k-way cut problem can be solved in $\widetilde{O}(mn^k)$ time.

Karger and Stein [11] and Nagamochi et al. [15] have studied the bounds on the number of small cuts, which motivates the following question: Can we give nontrivial lower and upper bounds on the number of minimum k-way cuts? It is easy to get a lower bound of $O(n^k)$. Note that in a cycle consisting of n edges with equal weight, the number of minimum k-way cuts is $\binom{n}{k}$. Can better bounds be achieved?

References

1. Burlet, M., Goldschmidt, O.: A new and improved algorithm for the 3-cut problem. Operations Research Letters 21(5), 225–227 (1997)
2. Dahlhaus, E., Johnson, D., Papadimitriou, C., Seymour, P., Yannakakis, M.: The complexity of multiterminal cuts. SIAM J. Comput. 23(4), 864–894 (1994)
3. Goldberg, A.V., Rao, S.: Beyond the flow decomposition barrier. J. ACM 45(5), 783–797 (1998)

4. Goldberg, A.V., Tarjan, R.E.: A new approach to the maximum-flow problem. J. ACM 35(4), 921–940 (1988)
5. Goldschmidt, O., Hochbaum, D.: A polynomial algorithm for the k-cut problem for fixed k. Mathematics of Operations Research 19(1), 24–37 (1994)
6. He, X.: An improved algorithm for the planar 3-cut problem. J. Algorithms 12(1), 23–37 (1991)
7. Ford, J.R., Fullkerson, D.R.: Flows in networks. Princeton University Press, Princeton (1962)
8. Kamidoi, Y., Wakabayashi, S., Yoshida, N.: A divide-and-conquer approach to the minimum k-way cut problem. Algorithmica 32(2), 262–276 (2002)
9. Kamidoi, Y., Yoshida, N., Nagamochi, H.: A deterministic algorithm for finding all minimum k-way cuts. SIAM Journal on Computing 36(5), 1329–1341 (2006)
10. Kapoor, S.: On minimum 3-cuts and approximating k-cuts using cut trees. In: Cunningham, W.H., Queyranne, M., McCormick, S.T. (eds.) IPCO 1996. LNCS, vol. 1084. Springer, Heidelberg (1996)
11. Karger, D.R., Stein, C.: A new approach to the minimum cut problem. Journal of the ACM 43(4), 601–640 (1996)
12. Nagamochi, H., Ibaraki, T.: Computing edge connectivity in multigraphs and capacitated graphs. SIAM Journal on Discrete Mathematics 5(1), 54–66 (1992)
13. Nagamochi, H., Ibaraki, T.: A fast algorithm for computing minimum 3-way and 4-way cuts. Mathematical Programming 88(3), 507–520 (2000)
14. Nagamochi, H., Katayama, S., Ibaraki, T.: A faster algorithm for computing minimum 5-way and 6-way cuts in graphs. In: Asano, T., Imai, H., Lee, D.T., Nakano, S.-i., Tokuyama, T. (eds.) COCOON 1999. LNCS, vol. 1627. Springer, Heidelberg (1999)
15. Nagamochi, H., Nishimura, K., Ibaraki, T.: Computing all small cuts in an undirected network. SIAM Journal on Discrete Mathematics 10(3), 469–481 (1997)
16. Naor, J., Rabani, Y.: Tree packing and approximating k-cuts. In: Proceedings of the twelfth annual ACM-SIAM symposium on discrete algorithms (SODA 2001). Society for Industrial and Applied Mathematics, Philadelphia (2001)
17. Ravi, R., Sinha, A.: Approximating k-cuts via network strength. In: Proceedings of the thirteenth annual ACM-SIAM symposium on Discrete algorithms (SODA 2002). Society for Industrial and Applied Mathematics, Philadelphia (2002)
18. Saran, H., Vazirani, V.V.: Finding k-cuts within twice the optimal. SIAM J. Comput. 24(1), 101–108 (1995)
19. Stoer, M., Wagner, F.: A simple min-cut algorithm. J. ACM 44(4), 585–591 (1997)
20. Thorup, M.: Minimum k-way cuts via deterministic greedy tree packing. In: Proceedings of the 40th Annual ACM Symposium on Theory of Computing (STOC 2008) (2008)
21. Xiao, M.: Algorithms for multiterminal cuts. In: Hirsch, E.A., Razborov, A.A., Semenov, A., Slissenko, A. (eds.) Computer Science – Theory and Applications. LNCS, vol. 5010. Springer, Heidelberg (2008)
22. Xiao, M.: Finding minimum 3-way cuts in hypergraphs. In: Agrawal, M., Du, D.-Z., Duan, Z., Li, A. (eds.) TAMC 2008. LNCS, vol. 4978. Springer, Heidelberg (2008)
23. Xiao, M., Cai, L., Yao, A.C.: Tight approximation ratio of a general greedy splitting algorithm for the minimum k-way cut problem (manuscript, 2007)
24. Zhao, L., Nagamochi, H., Ibaraki, T.: Approximating the minimum k-way cut in a graph via minimum 3-way cuts. J. Comb. Optim. 5(4), 397–410 (2001)