LP Relaxation and Tree Packing for Minimum k-cuts*

Chandra Chekuri† Kent Quanrud‡ Chao Xu§

November 5, 2018

Abstract

Karger used spanning tree packings [14] to derive a near linear-time randomized algorithm for the global minimum cut problem as well as a bound on the number of approximate minimum cuts. This is a different approach from his well-known random contraction algorithm [13, 15]. Thorup developed a fast deterministic algorithm for the minimum k-cut problem via greedy *recursive* tree packings [29].

In this paper we revisit properties of an LP relaxation for k-cut proposed by Naor and Rabani [21], and analyzed in [3]. We show that the dual of the LP yields a tree packing, that when combined with an upper bound on the integrality gap for the LP, easily and transparently extends Karger's analysis for mincut to the k-cut problem. In addition to the simplicity of the algorithm and its analysis, this allows us to improve the running time of Thorup's algorithm by a factor of n. We also improve the bound on the number of α-approximate k-cuts. Second, we give a simple proof that the integrality gap of the LP is $2(1-1/n)$. Third, we show that an optimum solution to the LP relaxation, for all values of k, is fully determined by the principal sequence of partitions of the input graph. This allows us to relate the LP relaxation to the Lagrangean relaxation approach of Barahona [2] and Ravi and Sinha [24]; it also shows that the idealized recursive tree packing considered by Thorup gives an optimum dual solution to the LP. This work arose from an effort to understand and simplify the results of Thorup [29].

1 Introduction

The global minimum cut problem in graphs (MinCut) is well-known and extensively studied. Given an undirected graph $G = (V, E)$ with non-negative edge capacities $c : E \to \mathbb{R}_+$, the goal is to remove a minimum capacity set of edges such that the residual graph has at least two connected components. When all capacities are one, the mincut of a graph is its global edge-connectivity. The k-Cut problem is a natural generalization. Given a graph $G = (V, E)$ and an integer $k \geq 2$, the goal is to remove a minimum capacity set of edges such that the residual graph has at least k connected components. MinCut and k-Cut have been extensively studied in the literature. Initial algorithms for MinCut were based on a reduction to the s-t-mincut problem. However, it was realized later on that it can be solved more efficiently and directly. Currently the best deterministic algorithm for MinCut runs in $O(mn + n^2 \log n)$ time [27] and is based on the maximum adjacency ordering approach of Nagamochi and Ibaraki [19]. On the other hand, there is a near-linear time Monte Carlo randomized algorithm due to Karger [14]. Bridging the gap between the running times for the deterministic and randomized algorithms is a major open problem. In recent work [12, 16] obtained near-linear time deterministic algorithms for *simple* unweighted graphs.

*Work on this paper supported in part by NSF grant CCF-1526799.

†Department of Computer Science, University of Illinois, Urbana, IL 61801. chekuri@illinois.edu.

‡Department of Computer Science, University of Illinois, Urbana, IL 61801. quanrud2@illinois.edu.

§Yahoo! Research, New York, NY 10003. chao.xu@oath.com. Work done while the author was at University of Illinois.

The k-CUT problem is NP-Hard if k is part of the input [10], however, there is a polynomial-time algorithm for any fixed k. Such an algorithm was first devised by Goldschmidt and Hochbaum [10], and subsequently there have been several different algorithms improving the run-time. The randomized algorithm of Karger and Stein [15] runs in $\tilde{O}(n^{2(k-1)})$ time and outputs the optimum cut with high probability. The fastest deterministic algorithm, due to Thorup [29], runs in $\tilde{O}(mn^{2k-2})$ time [29]. Recent work of Gupta, Lee and Li [11] obtains a faster run-time of $\tilde{O}(k^{O(k)}n^{(2\omega/3+o(1))k})$ if the graph has small integer weights, where ω is the exponent in the run-time of matrix multiplication. It is also known that k-CUT is $W[1]$-hard when parameterized by k [6]; that is, we do not expect an algorithm with a run-time of $f(k)n^{O(1)}$. Several algorithms that yield a 2-approximation are known for k-CUT; Saran and Vazirani's algorithm based on repeated minimum-cut computations gives $(2-2/k)$-approximation [25]; the same bound can be achieved by removing the $(k-1)$ smallest weight edges in a Gomory-Hu tree of the graph [25]. Nagamochi and Kamidoi showed that using the concept of extreme sets, a $(2-2/k)$-approximation can be found even faster [20]. Naor and Rabani developed an LP relaxation for k-CUT [21] and this yields a $2(1-1/n)$-approximation [3]. Ravi and Sinha [24] obtained another $2(1-1/n)$-approximation via a Lagrangean relaxation approach which was also considered independently by Barahona [2]. A factor of 2, for large k, is the best possible approximation under the Small Set Expansion hypothesis [18]. Recent work has obtained a 1.81 approximation in $2^{O(k^2)}n^{O(1)}$ time [11]; whether a PTAS can be obtained in $f(k)\text{poly}(n)$ time is an interesting open problem.

Motivation and contributions: The main motivation for this work was to simplify and understand Thorup's tree packing based algorithm for k-CUT. Karger's near-linear time algorithm and analysis for the MINCUT problem [14] is based on the well-known theorem of Tutte and Nash-Williams (on the minmax relation for edge-disjoint trees in a graph). It is simple and elegant; the main complexity is in the improved running time which is achieved via a complex dynamic program. Karger also tightened the bound on the number of α-approximate minimum cuts in a graph (originally shown via his random contraction algorithm) via tree packings. In contrast to the case of mincut, the main structural result in Thorup's work on k-CUT is much less easy to understand and motivate. His proof consists of two parts. He shows that an ideal tree packing obtained via a recursive decomposition of the graph, first outlined in [28], has the property that any optimum k-cut crosses some tree in the packing at most $2k-2$ times. The second part argues that a greedy tree packing with sufficiently many trees approximates the ideal tree packing arbitrarily well. The greedy tree packing is closely related to a multiplicative weight update method for solving a basic tree packing linear program, however, no explicit LP is used in Thorup's analysis. Thus, although Thorup's algorithm is very simple to describe (and implement), the analysis is somewhat opaque.

In this paper we make several contributions which connect Thorup's tree packing to the LP relaxation for k-CUT [21]. We outline the specific contributions below.

- We show that the dual of the LP for k-CUT gives a tree packing and one can use a simple analysis, very similar to that of Karger, to show that any optimum k-cut crosses some tree in the packing at most $(2k-3)$ times. Thorup proved a bound of $(2k-2)$ for his tree packing. This leads to a slightly faster algorithm than that of Thorup and also to an improved bound on the number of approximate k-cuts.

- We give a new and simple proof that the integrality gap of the LP for k-cut is upper bounded by $2(1-1/n)$. We note that the proof claimed in [21] was incorrect and the proof in [3] is indirect and technical.

- We show that the optimum solution of the k-cut LP, for all values of k, can be completely characterized by the principal sequence of partitions of the cut function of the given graph. This establishes the connection between the dual of the LP relaxation and the ideal recursive tree

packing considered by Thorup. It also shows that the lower bound provided by the LP relaxation is equivalent to the Lagrangean relaxation lower bound considered by Barahona [2] and Ravi and Sinha [24].

Our results help unify and simplify the different approaches to k-cut via the LP relaxation and its dual. A key motivation for this paper is to simplify and improve the understanding of the tree packing approach. For this reason we take a leisurely path and reprove some of Karger's results for the sake of completeness, and to point out the similarity of our argument for k-CUT to the case of MINCUT. Readers familiar with [14] may wish to skip section 3.

Organization: section 2 sets up some basic notation and definitions. section 3 discusses Karger's approach for MINCUT via tree packings with some connections to recent developments on approximately solving tree packings. section 4 describes the tree packing obtained from the dual of the LP relaxation for k-CUT and how it can be used to extend Karger's approach to k-CUT. section 5 gives a new proof that the LP integrality gap for k-CUT is $2(1-1/n)$. In section 6 we show that the optimum LP solution for all values of k can be characterized by a recursive decomposition of the input graph.

2 Preliminaries

We use n and m to denote the number of nodes and edges in a given graph. For a graph $G = (V, E)$, let $\mathcal{T}(G)$ denote the set of spanning trees of G. For a graph $G = (V, E)$ with edge capacities $c : E \to \mathbb{R}_+$ the *fractional spanning tree packing number*, denoted by $\tau(G)$, is the optimum value of a simple linear program shown in Figure 2.1 whose variables are $y_T, T \in \mathcal{T}(G)$. The LP has an exponential number of variables but is still polynomial time solvable. There are several ways to see this and efficient strongly combinatorial algorithms are also known [8]. We also observe that there is an optimum solution to the LP whose support has at most m trees since the number of non-trivial constraints in the LP is at most m (one per each edge).

$$\begin{aligned} \max \sum_{T \in \mathcal{T}(G)} & y_T \\ \sum_{T \ni e} & y_T \leq c(e) \quad e \in E \\ & y_T \geq 0 \qquad T \in \mathcal{T}(G) \end{aligned}$$

Figure 2.1: LP relaxation defining $\tau(G)$.

There is a min-max formula for $\tau(G)$ which is a special case of the min-max formula for matroid base packing due to Tutte and Nash-Williams. To state this theorem we introduce some notation. For a partition $\mathcal{P}$ of the vertex set V let $E(\mathcal{P})$ denote the set of edges that cross the partition (that is, have end points in two different parts) and let $|\mathcal{P}|$ denote the number of parts of $\mathcal{P}$. A *k-cut* is $E(\mathcal{P})$ for some partition $\mathcal{P}$ such that $|\mathcal{P}| \geq k$. A *cut* is a 2-cut. The value of the minimum cut of G is denoted as $\lambda(G)$. It is not hard to see that for any partition $\mathcal{P}$ of the vertex set V, $\tau(G) \leq \frac{c(E(\mathcal{P}))}{|\mathcal{P}|-1}$ since every spanning tree of G contains at least $|\mathcal{P}|-1$ edges from $E(\mathcal{P})$. The minimum over all partitions of the quantity, $\frac{c(E(\mathcal{P}))}{|\mathcal{P}|-1}$, is also referred to as the *strength* of G (denoted by $\sigma(G)$), and turns out to be equal to $\tau(G)$.

Theorem 2.1 (Tutte and Nash-Williams) *For any undirected edge capacitated graph G,*

$$\tau(G) = \min_{\mathcal{P}} \frac{c(E(\mathcal{P}))}{|\mathcal{P}|-1}.$$

Tutte and Nash-Williams proved the integer packing version of the preceding theorem which is harder; they showed that the maximum number of edge-disjoint spanning trees in a graph G with integer capacities c is give by $\min_{\mathcal{P}} \lfloor \frac{c(E(\mathcal{P}))}{|\mathcal{P}|-1} \rfloor$. The theorem is in fact a special case of matroid base packing theorem and can also be derived via the matroid union theorem of Edmonds; we refer the reader to [26].

A useful and well-known corollary of the preceding theorem is given below.

Corollary 2.2 *For any graph G, $\tau(G) \geq \frac{n}{2(n-1)} \cdot \lambda(G)$. If G is an unweighted graph then $\tau(G) \geq \frac{\lambda(G)+1}{2}$.*

Proof: Consider the partition $\mathcal{P}^*$ that achieves the minimum in the minmax formula. We have $c(E(\mathcal{P})) \geq |\mathcal{P}^*|\lambda(G)/2$ since the capacity of edges leaving each part of $\mathcal{P}^*$ is at least $\lambda(G)$ and an edge in $E(\mathcal{P}^*)$ crosses exactly two parts. Thus,

$$\tau(G) = \frac{c(E(\mathcal{P}^*))}{|\mathcal{P}^*|-1} \geq \frac{|\mathcal{P}^*|\lambda(G)}{2(|\mathcal{P}^*|-1)} \geq \frac{n}{2(n-1)}\lambda(G)$$

since $|\mathcal{P}^*| \leq n$. If G is unweighted graph then $|\mathcal{P}| \leq \lambda(G)+1$ and hence $\tau(G) \geq \frac{\lambda(G)+1}{2}$ as desired. □

We say that a tree packing $y : \mathcal{T}(G) \to \mathbb{R}_+$ is $(1-\epsilon)$-approximate if $\sum_{T \in \mathcal{T}(G)} y_T \geq (1-\epsilon)\tau(G)$. Note that we typically want a compact tree packing that can either be explicitly specified via a small number of trees or even implicitly via a data structure representing a collection of trees. Approximate spanning tree packings have been obtained via greedy spanning tree packings which can be viewed as applying the multiplicative weight update method. Recently [4] obtained the following result.

Theorem 2.3 ([4]) *There is a deterministic algorithm that, given an edge-capacitated undirected graph on m edges and an $\epsilon \in (0, 1/2)$, runs in $O(m \log^3 n/\epsilon^2)$ time and outputs an implicit representation of a $(1-\epsilon)$-approximate tree packing.*

3 Tree packings and minimum cuts

We review some of Karger's observations and results connecting tree packings and minimum cuts [14] which follow relatively easily via Corollary 2.2. In this section, we restrict the definition of a cut to an edge set $E(\mathcal{P})$ for a partition $\mathcal{P} = \{S, V \setminus S\}$ with *exactly* two parts. Hence, we use $\delta(S)$ to uniquely identify a cut. However, as one can see later in section 4, the results will also hold for the more general definition of a cut, where $\mathcal{P}$ have *at least* two parts. We rephrase his results and arguments with a slightly different notation. Given a spanning tree T and a cut $A \subseteq E$, following Karger, we say that T h-respects A for some integer $h \geq 1$ if $|E(T) \cap A| \leq h$.

Karger proved that a constant fraction of trees (in the weighted sense) of an optimum packing 2-respect any fixed mincut. In fact this holds for a $(1-\epsilon)$-approximate tree packing for sufficiently small ϵ. The proof, as follows, is an easy consequence of Corollary 2.2 and an averaging argument. It is convenient to view a tree packing as a probability distribution. Let $p_T = y_T/\tau(G)$. We then have $\sum_T p_T = 1$ for an exact tree packing and for a $(1-\epsilon)$-packing we have $\sum_T p_T \in (1-\epsilon, 1]$. Let $\delta(S)$ be a fixed minimum cut whose capacity is $\lambda(G)$. Let $\ell_T = |E(T) \cap \delta(S)|$ be the number of edges of T that cross S. Let $q = \sum_{T:\ell_T \leq 2} p_T$ be the fraction of trees that 2-respect $\delta(S)$. Since each tree crosses S at least once we have,

$$\sum_T p_T \ell_T = \sum_{T:\ell_T \leq 2} p_T \ell_T + \sum_{T:\ell_T \geq 3} p_T \ell_T \geq q + 3(1-\epsilon-q).$$

Because y is a valid packing,

$$\tau(G)\sum_T p_T \ell_T = \sum_T y_T \ell_T \leq c(\delta(S)) = \lambda(G).$$

Putting the two inequalities together and using Corollary 2.2,

$$3(1-\epsilon)-2q \leq \lambda(G)/\tau(G) \leq 2(n-1)/n$$

which implies that

$$q \geq \frac{3}{2}(1-\epsilon)-(1-1/n) = \frac{1}{2}+\frac{1}{n}-\frac{3\epsilon}{2}.$$

If $\epsilon = 0$ this implies that at least half the fraction of trees 2-respect any minimum cut. Let q' be the fraction of trees that 1-respect a minimum cut. One can do similar calculations as above to conclude that

$$q' \geq 2(1-\epsilon)-2(1-1/n) \geq 2(\frac{1}{n}-\epsilon).$$

Thus, $q' > 0$ as long as $\epsilon < 1/n$. In an optimum packing there is always a tree in the support that 1-respects a mincut. The preceding argument can be generalized in a direct fashion to yield the following useful lemma on α-approximate cuts.

Lemma 3.1 *Let y be a $(1-\epsilon)$-approximate tree packing. Let $\delta(S)$ be an α-approximate minimum cut (i.e., $c(\delta(S)) \leq \alpha\lambda(G)$) for some fixed $\alpha \geq 1$. For a fixed integer $h \geq 1$, let q_h denote the fraction of trees in the packing y that h-respect $\delta(S)$. Then*

$$q_h \geq (1-\epsilon)\left(1+\frac{1}{h}\right)-\frac{2\alpha}{h}\left(1-\frac{1}{n}.\right)$$

3.1 Number of approximate minimum cuts

Karger showed that the number of α-approximate minimum cuts is at most $O(n^{2\alpha})$ via his random contraction algorithm [13]. He improved the bound to $O(n^{\lfloor 2\alpha \rfloor})$ (for any fixed α) via tree packings in [14]. We review the latter argument.

For any cut $\delta(S)$, we associate the subset of edges of T that cross S, $E(T) \cap \delta(S)$. In the other direction, removing a set of edges $A \subseteq E(T)$ induces several components in $T-A$, which induces a unique cut in G where any two components of $T-A$ adjacent in T lie on opposite sides of the cut. This gives a bijection between cuts induced by edge removals in T, and cuts in the graph.

Fix $\alpha > 1$, and let $h = \lfloor 2\alpha \rfloor$. Let y be a fixed optimum tree packing supported by some $m' \leq m$ trees. For any α-approximate mincut $\delta(S)$, by Lemma 3.1 and some simplification, the fraction of trees in the packing y that h-respects $\delta(S)$ is at least

$$q_{h,\alpha} \equiv \frac{1}{\lfloor 2\alpha \rfloor}\left(1-(2\alpha-\lfloor 2\alpha \rfloor)(1-1/n)\right).$$

Observing that $q_{h,\alpha} > 0$, an easy counting argument for approximate mincuts is the following. For each α-approximate mincut, there is at least one tree in the (support of the) packing y which crosses it at most h times. Hence each α-approximate cut can be mapped to a distinct combination of a tree in the packing and at most h edges from that tree. The total number of these combinations is $m'\binom{n-1}{h} = O(mn^h)$.

We can avoid the factor of m by leveraging the fact that $q_{h,\alpha}$ is a constant for every fixed α. We give an informal argument here. A tree can h-respect at most $h^h\binom{n-1}{h}$ distinct cuts, while each α-approximate minimum cut is h-respected by a (constant) $q_{h,\alpha}$-fraction of the tree packing. It follows by a packing argument that the total number of α-approximate mincuts is at most

$$\frac{h^h\binom{n-1}{h}}{q_{h,\alpha}} = O\left(n^h\right).$$

3.2 Algorithm for minimum cut via tree packings

Karger used tree packings to obtain a randomized near linear time algorithm for the global minimum cut. The algorithm is based on combining the following two steps.

- Given a graph G there is a randomized algorithm that outputs $O(\log n)$ trees in $\tilde{O}(m)$ time such that with high probability there is a global minimum cut that 2-respects one of the trees in the packing.
- There is a deterministic algorithm that given a graph G and a spanning tree T, in $\tilde{O}(m)$ time finds the cut of minimum capacity in G that 2-respects T. This is based on a clever dynamic programming algorithm that utilizes dynamic tree data structures.

Only the first step of the algorithm is randomized. Karger solves the first step as follows. Given a capacitated graph G and an $\epsilon > 0$, he sparsifies the graph G to obtain an unweighted skeleton graph H via random sampling such that (i) H has $O(n \log n/\epsilon^2)$ edges (ii) $\lambda(H) = \Theta(\log n/\epsilon^2)$ and (iii) a minimum cut of G corresponds to a $(1+\epsilon)$-approximate minimum cut of H in that the cuts induce the same vertex partition. Karger then uses greedy tree packing in H to obtain a $(1-\epsilon')$-tree packing in H with $O(\log n/\epsilon'^2)$ trees, and via Corollary 2.2 argues that one of the trees in the packing 2-respects a mincut of G; here ϵ and ϵ' are chosen to be sufficiently small but fixed constants.

We observe that Theorem 2.3 can be used in place of the sparsification step of Karger. The deterministic algorithm implied by the theorem can be used to find an implicit $(1-\epsilon)$-approximate tree packing in near linear time for any fixed $\epsilon > 0$. For sufficiently small but fixed ϵ, a constant fraction of the trees in the tree packing 2-respect any fixed minimum cut. Thus, if we sample a tree T from the tree packing, and then apply Karger's deterministic algorithm for finding the smallest cut that 2-respects T, we can find a minimum cut with constant probability. We can repeat the sampling $\Theta(\log n)$ times to obtain a high probability bound.

Karger raised the following question in his paper. Can the dynamic programming algorithm for finding the minimum cut that 2-respects a tree be made *dynamic*? That is, suppose T is altered via edge swaps to yield a tree $T' = T - e + e'$ where $e \in E(T)$ is removed and replaced by a new edge e'. Can the solution for T be updated quickly to obtain a solution for T'? Note that G is static, only the tree is changing. The tree packing from Theorem 2.3 finds an implicit packing via $\tilde{O}(m)$ edge swap operations from a starting tree T_0. Suppose there is a dynamic version of Karger's dynamic program that handles updates to the tree in amortized $g(n)$ time per update. This would yield a *deterministic* algorithm for the global mincut with a total time of $\tilde{O}(mg(n))$. We note that the best deterministic algorithm for capacitated graphs is $O(mn + n^2 \log n)$ [27]. This would be improved be any $g(n) = o(n)$.

4 Tree packings for k-Cut via the LP relaxation

In this section we consider the k-Cut problem. Thorup [28] constructed a probability distribution over spanning trees which were obtained via a recursive greedy tree packing and showed that there is a tree T in the support of the distribution such that a minimum weight k-cut contains at most $2(k-1)$ edges of T. He then showed that greedy tree packing with $O(mk^3 \log n)$ trees closely approximates the ideal distribution. With this approach, he derived the currently fastest known deterministic algorithm to find the minimum k-Cut in $\tilde{O}(mn^{2k-2})$ time. This is only slightly slower than the randomized Monte Carlo algorithm of Karger and Stein [15] whose algorithm runs in $\tilde{O}(n^{2k-2})$ time. Thorup's algorithm is fairly simple. However, the proofs are somewhat complex since they rely on the recursive tree packing and its subtle properties. Arguing that the greedy tree packing approximates the recursive tree packing is also technical.

Here we consider a different tree packing for k-Cut that arises from the LP relaxation for k-Cut considered by Naor and Rabani [21]. This LP relaxation is shown in Figure 4.1. The variables are

$$\begin{aligned}
\min \quad & \sum_{e \in E)} c_e x_e \\
\text{s.t.} \quad & \sum_{e \in T} x_e \geq k-1 \text{ for all } T \in \mathcal{T}(G) \\
& x_e \leq 1 \text{ for all } e \in E \\
& x_e \geq 0 \text{ for all } e \in E
\end{aligned}$$

Figure 4.1: An LP relaxation for the k-CUT problem from [21].

$x_e \in [0,1], e \in E$ which indicate whether an edge e is cut or not. There is a constraint for each spanning tree $T \in \mathcal{T}(G)$; at least $k-1$ edges from T need to be chosen in a valid k-cut. We note that for $k > 2$ the upper bound constraint $x_e \leq 1$ is necessary.

The dual of the LP is given in Figure 4.2. Naor and Rabani claimed an integrality gap of 2 for the k-CUT LP. Their proof was incomplete and a correct proof was given in [3] in the context of a more general problem called the Steiner k-CUT problem. Let $\lambda_k(G)$ denote the minimum k-cut capacity in G.

Theorem 4.1 ([3]) *The worst case integrality gap of the LP for k-CUT in Figure 4.1 is $2(1-1/n)$.*

Corollary 4.2 *Let (y,z) be an optimum solution for the dual LP for k-CUT shown in Figure 4.2. Then*

$$(k-1)\sum_T y_T \geq \frac{n\lambda_k(G)}{2(n-1)} + z(E).$$

Note that Corollary 2.2 is a special case of the preceding corollary.

Remark We note that the LP relaxation in Figure 4.1 assumes that G is connected. This is easy to ensure by adding dummy edges of zero cost to make G connected. However, it is useful to consider the general case when the number of connected components in G is h where we assume for simplicity that $h < k$ (if $h \geq k$ the problem is trivial). In this case we need to consider the maximal forests in G, each of which has exactly $n-h$ edges; to avoid notational overload we use $\mathcal{T}(G)$ to denote the set of maximal forests of G. The LP constraint now changes to

$$\sum_{e \in T} x_e \geq k-h \quad T \in \mathcal{T}(G).$$

Tree packing interpretation of the dual LP: The dual LP has two types of variables. For each edge e there is a variable z_e and for each spanning tree T there is a variable y_T. The dual seeks to add capacity $z : E \to \mathbb{R}_+$ to the original capacities c, and then find a maximum tree packing $y : \mathcal{T}(G) \to \mathbb{R}_+$ within the augmented capacities $c + z$. The objective is $(k-1)\sum_T y_T - \sum_{e \in E} z_e$. Note that for $k = 2$, there is an optimum solution with $z = 0$; this can be seen by the fact that for $k = 2$ the primal LP can omit the constraints $x_e \leq 1$ for $e \in E$. For $k > 2$ it may be advantageous to add capacity to some bottleneck edges (say from a minimum cut) to increase the tree packing value, which is multiplied by $(k-1)$.

Our goal is to show that one can transparently carry over the arguments for global minimum cut via tree packings to the k-CUT setting via (optimum) solutions y, z to the dual LP. Theorem 4.1 plays the role of Corollary 2.2. The key lemma below is analogous to Lemma 3.1.

$$\begin{aligned} \max\ & (k-1)\sum_{T\in\mathcal{T}(G)} y_T - \sum_{e\in E} z_e \\ \text{s.t.}\ & \sum_{T\ni e} y_T \le c_e + z_e \text{ for all } e\in E \\ & y_T \ge 0 \text{ for all } T\in\mathcal{T}(G) \end{aligned}$$

Figure 4.2: Dual of the LP relaxation from Figure 4.1.

Lemma 4.3 *Let y, z be an optimum solution to the dual LP for k-Cut shown in Figure 4.2. Let $E' \subseteq E$ be any subset of edges such that $c(E') \le \alpha\lambda_k(G)$ for some $\alpha \ge 1$. For each tree T, let $\ell_T = |E' \cap E(T)|$ denote the number of edges in both T and E'. For an integer $h \ge (k-1)$, let $q_h = \sum_{T:\ell_T \le h} p_T$ denote the fraction of the trees in the packing induced by y, z that contain at most h edges from E'. Then*

$$q_h \ge 1 - \frac{2\alpha(k-1)(1-\frac{1}{n})}{h+1}.$$

Proof: Let $\tau_k(G)$ denote $\sum_T y_T$ and let $p_T = y_T/\tau_k(G)$. Thus,

$$\sum_T p_T\ell_T = \sum_{T:\ell_T\le h} p_T\ell_T + \sum_{T:\ell_T\ge(h+1)} p_T\ell_T \ge (h+1)(1-q_h).$$

Because y is a valid tree packing in capacities $c+z$,

$$\tau_k(G)\sum_T p_T\ell_T = \sum_T y_T\ell_T \le c(E') + z(E') \le \alpha\lambda_k(G) + z(E') \le \alpha(\lambda_k(G) + z(E)).$$

In the second to last inequality uses the fact that $\alpha \ge 1$ and $z \ge 0$. Putting the preceding inequalities together, we have

$$(h+1)(1-q_h) \le \frac{1}{\tau_k(G)}\alpha(\lambda_k(G) + z(E)).$$

We rearrange and simplify the inequality in Corollary 4.2 as

$$2\left(1-\frac{1}{n}\right)(k-1)\sum_T y_T \ge \lambda_k(G) + 2(1-1/n)z(E) \ge \lambda_k(G) + z(E).$$

Plugging this inequality into the preceding one yields

$$(h+1)(1-q_h) \le 2\alpha(k-1)(1-1/n),$$

which implies that

$$q_h \ge 1 - \frac{2\alpha(k-1)(1-\frac{1}{n})}{h+1}.$$

□

Remark Lemma 4.3 does not require A to be a k-cut. Ultimately we will only apply Lemma 4.3 to k-cuts.

Corollary 4.4 *Let (y,z) be an optimum solution to the dual LP. For every optimum k-cut $A \subseteq E$ there is a tree T in the support of y such that $|E(T) \cap A| \leq 2k-3$.*

Proof: We apply Lemma 4.3 with $h = 2k-3$ and $\alpha = 1$ and observe that $q_h > 0$, which implies the desired statement. □

Corollary 4.5 *Let (y,z) be a $(1-\epsilon)$-approximate solution to the dual LP where $\epsilon < \frac{1}{2k-1}$. For every optimum k-cut $A \subseteq E$ there is a tree T in the support of y such that $|E(T) \cap A| \leq 2k-2$.*

Proof: Let q_h be defined as in Lemma 4.3. If (y,z) is a $(1-\epsilon)$-approximate solution to the dual LP, we would have

$$(k-1)\sum_T y_T \geq (1-\epsilon)\frac{n\lambda_k(G)}{2(n-1)} + z(E) \geq (1-\epsilon)\frac{\lambda_k(G)}{2} + z(E) \tag{1}$$

in place of Corollary 4.2.

Examining the proof of Lemma 4.3, we see that optimality of (y,z) is not used in the proof except when invoking Corollary 4.2. Repeating the proof of Lemma 4.3, except using (1) instead of Corollary 4.2 and setting $\alpha = 1$, we obtain the bound

$$q_h \geq 1 - \frac{2(k-1)}{(h+1)(1-\epsilon)}.$$

We observe that $q_h > 0$ for $h = 2k-2$ and $\epsilon < \frac{1}{2k-1}$. □

4.1 Number of approximate k-cuts

We now prove the following theorem.

Theorem 4.6 *Let $G = (V,E)$ be an undirected edge-weighted graph and let k be a fixed integer. For $\alpha \geq 1$, the number of cuts with weight $\leq \alpha\lambda_k(G)$ is $O\left(n^{\lfloor 2\alpha(k-1)\rfloor}\right)$.*

Proof: Let $h = \lfloor 2\alpha(k-1)\rfloor$. By Lemma 4.3, there is a fixed value

$$q_h = 1 - \frac{2\alpha(k-1)\left(1-\frac{1}{n}\right)}{h+1} > 0$$

such that for any cut A with total weight $\leq \alpha\lambda_k(G)$, then at least a q_h-weighted fraction of trees in the tree packing y contains at most h edges of A.

For a given tree T, consider the number of distinct cuts that contain h or fewer edges in T. There are at most n^h subsets of the tree's edges of size at most h, and each selection induces $f(h)$ partitions of the components $\leq h+1$ into at least 2 parts for some $f(h) < h^h$. Thus there are at most $f(h)n^h$ cuts containing h or fewer edges from T for some fixed function f.

If there are (strictly) more than $f(h)n^h/q_h$ distinct cuts with weight at most $\alpha\lambda_k(G)$, then by the pigeonhole principle there exists a tree T in the packing that induces strictly more than $f(h)n^h$ different cuts with h or fewer edges – a contradiction. □

Like Lemma 4.3, Theorem 4.6 is not restricted to k-cuts. The primary application of Theorem 4.6 is to count the number of approximate minimum k-cuts, as follows.

Corollary 4.7 *Let $G = (V,E)$ be an undirected edge-weighted graph and let k be a fixed integer. For $\alpha \geq 1$, the number of α-approximate minimum k-cuts is $O(n^{\lfloor 2\alpha(k-1)\rfloor})$.*

4.2 Enumerating all minimum k-cuts

We briefly describe how to enumerate all k-cuts via Lemma 4.3. The argument is basically the same as that of Karger and Thorup. First, we compute an optimum solution (y^*, z^*) to the dual LP. We can do this via the Ellipsoid method or other ways. Let $\beta(n,m)$ be the running time to find (y^*, z^*). Moreover, we find a basic feasible solution to the dual LP we are guarantees that the support of y has at most m distinct trees. Now Lemma 4.3 guarantees that for every minimum k-cut $A \subseteq E$ there is a tree T such that $y(T) > 0$ and T $(2k-3)$-respects A. Thus, to enumerate all minimum k-cuts the following procedure suffices. For each of the trees T in the optimum packing we enumerate all k-cuts induced by removing $h = 2k-3$ edges from T. With appropriate care and data structures (see [14] and [28]) this can be done for a single tree T in $\tilde{O}(n^{2k-3} + m)$ time. The total time over all m trees in the support of y is $\tilde{O}(mn^{2k-3})$ for $k > 2$. This gives the following theorem.

Theorem 4.8 *For $k > 2$ all the minimum k-cuts of a graph with n nodes and m edges can be computed in time $\tilde{O}(mn^{2k-3} + \beta(m,n))$ time where $\beta(m,n)$ is the time to find an optimum solution to the LP for k-cut.*

We observe that Thorup's algorithm [28] runs in time $\tilde{O}(mn^{2k-2})$. Thorup uses greedy tree packing in place of solving the LP. The optimality of the LP solution was crucial in using the bound of $2k-3$ instead of $2k-2$. Thus, even though we obtain a slightly faster algorithm than Thorup, we need to find an optimum solution to the LP which can be done via the Ellipsoid method. The Ellipsoid method is not quite practical. Below we discuss a different approach.

In recent work Quanrud showed that a $(1-\epsilon)$-approximate solution to the dual LP can be computed in near-linear time. We state his theorem below.

Theorem 4.9 ([23]) *There is an algorithm that computes a $(1-\epsilon)$-approximate solution (y,z) the dual LP in $O(m \log^3 n/\epsilon^2)$ time.*

We observe that the preceding theorem guarantees $O(m \log^3 n/\epsilon^2)$ trees in the support of y and also implicity stores them in $O(m \log^3 n/\epsilon^2)$ space. If we choose $\epsilon < 1/(2k-1)$ then, via Corollary 4.5, for every minimum k-cut $A \subseteq E$ there is a tree T in the support of y that $(2k-2)$-respects A. This leads to an algorithm that in $\tilde{O}(mn^{2k-2})$ time enumerates all minimum k-cuts and recovers Thorup's running time. However, we note that the trees generated by the algorithm in the preceding theorem are implicit, and can be stored in small space. It may be possible to use this additional structure to match or improve the run-time achieved by Theorem 4.8.

Remark For unweighted graphs with $\tilde{O}(\frac{m}{n-k}\frac{1}{\epsilon^2})$ trees [23] guarantees a $(1-\epsilon)$-approximation. This improves the running time to $\tilde{O}(mn^{2k-3})$ for unweighted graphs.

We briefly discuss a potential approach to speed up the computation futher. Recall that Karger describes an algorithm that given a spanning tree T of a graph G finds the minimum cut that 2-respects T in $\tilde{O}(m)$ time, speeding up the easier $\tilde{O}(n^2)$ time algorithm. We can leverage this as follows. In the case of $k > 2$ we are given T and G and wish to find the minimum k-cut induced by the removal of at most t edges where t is either $2k-3$ or $2k-2$ depending on the tree packing we use. Suppose A is a set of $t-2$ edges of T. Removing them from T yields a forest with $t-1$ components. We can then apply Kargers algorithm in each of these components with an appropriate graph. This results in a running time of $\tilde{O}(mn^{t-2})$ per tree rather than $\tilde{O}(n^t)$. We can try to build on Karger's ideas to improve the running time to find the best 3-cut induced by removing at most 4 edges from T. We can then leverage this for larger values of k.

5 A new proof of the LP integrality gap for k-CUT

The proof of Theorem 4.1 in [3] is based on the primal-dual algorithm and analysis of Agarwal, Klein and Ravi [1], and Goemans and Williamson [9] for the Steiner tree problem. For this reason the proof is technical and indirect. Further, the proof from [3] is described for the Steiner k-cut problem which has additional complexity. Here we give a different and intuitive proof for k-CUT. Unlike the proof in [3], the proof here relies on optimality properties of the LP solution and hence is less useful algorithmically. We note that [23] uses Theorem 4.9 and a fast implementation of the algorithmic proof in [3] to obtain a near-linear time $(2+\epsilon)$-approximation for k-CUT.

Let $G = (V, E)$ be a graph with non-negative edge capacities $c_e, e \in E$. We let $\deg(v) = \sum_{e \in \delta(v)} c_e$ denote the capacitated degree of node v. We will assume without loss of generality that $V = \{v_1, v_2, \ldots, v_n\}$ and that the nodes are sorted in increasing order of degrees, that is, $\deg(v_1) \leq \deg(v_2) \leq \ldots \leq \deg(v_n)$. We observe that $\deg(v_1) + \deg(v_2) + \ldots + \deg(v_{k-1})$ is an upper bound on the value of an optimum k-CUT; removing all the edges incident to $v_1, v_2, \ldots, v_{k-1}$ gives a feasible solution in which the components are the $k-1$ isolated vertices $\{v_1\}, \{v_2\}, \ldots, \{v_{k-1}\}$, and a component consisting of the remaining nodes of the graph.

The key lemma is the following which proves the integrality gap in a special case.

Lemma 5.1 *Let G be a connected graph and let x^* be an* optimum *solution to the k-CUT LP such that $x^*(e) \in (0,1)$ for each $e \in E$ (in other words x^* is fully fractional). Then $\sum_{i=1}^{k-1} \deg(v_i) \leq 2(1-1/n) \sum_e c_e x^*_e$.*

Proof: Let (y^*, z^*) be any fixed optimum solution to the dual LP. Complementary slackness gives the following two properties:

- $z^*(e) = 0$ for each $e \in E$, for if $z^*(e) > 0$ we would have $x^*(e) = 1$.
- for each $e \in E$, $\sum_{T \ni e} y^*_T = c_e$ since $x^*(e) > 0$.

From the second property above, and the fact that each spanning tree has exactly $(n-1)$ edges, we conclude that

$$(n-1) \sum_T y^*_T = \sum_{e \in E} c_e. \tag{2}$$

Since the degrees are sorted,

$$\sum_{i=1}^{k-1} \deg(v_i) \leq \frac{k-1}{n} \sum_{i=1}^{n} \deg(v_i) = 2\frac{k-1}{n} \sum_e c_e. \tag{3}$$

Putting the two preceding inequalities together,

$$\sum_{i=1}^{k-1} \deg(v_i) \leq 2(1-\frac{1}{n})(k-1) \sum_T y^*_T = 2(1-\frac{1}{n}) \sum_e c_e x^*_e,$$

where, the last equality is based on strong duality and the fact that $z^* = 0$. □

The preceding lemma can be easily generalized to the case when G has h connected components following the remark in the preceding section on the k-CUT LP. This gives us the following.

Corollary 5.2 *Let G be a graph with h connected components and let x^* be an* optimum *solution to the k-CUT LP such that $x^*(e) \in (0,1)$ for each $e \in E$. Then $\sum_{i=1}^{k-h} \deg(v_i) \leq 2(1-1/n) \sum_e c_e x^*_e$.*

Now we consider the general case when the optimum solution x^* to the k-CUT LP is not necessarily fully fractional as needed in Lemma 5.1. The following claim is easy.

Claim 5.1 *Let $x^*(e) = 0$ where $e = uv$. Let G' be the graph obtained from G by contracting u and v into a single node. Then there is a feasible solution x' to the k-CUT LP in G' of the same cost as that of x^*. Moreover a feasible k-cut in G' is a feasible k-cut in G of the same cost.*

Using the preceding claim we can assume without loss of generality that $x^*(e) > 0$ for each $e \in E$. Let $F = \{e \in E \mid x^*(e) = 1\}$. Since the LP solution x^* paid for the full cost of the edges in F, we can recurse on $G' = G - F$ and the fractional solution x' obtained by restricting x^* to $E \setminus F$. If G' is connected then x' is an optimum solution the k-CUT LP on G', and is fully fractional, and we can apply Lemma 5.1. However, G' can be disconnected. Let h be the number of connected components in G'. If $h \geq k$ we are done since A is a feasible k-cut and $c(A) \leq \sum_e c_e x^*_e$. The interesting case is when $2 \leq h < k$. In this case we apply Corollary 5.2 based on the following claim which is intuitive and whose formal proof we omit.

Claim 5.2 *Let x' be the restriction of x^* to $E \setminus F$. Then for any maximal forest T in G' we have $\sum_{e \in T} x'(e) \geq k - h$. Moreover, x' is an optimum solution to the k-CUT LP in G'.*

From Corollary 5.2 we can find $E' \subseteq E \setminus F$ such that $G' - E'$ induces a k-cut in G' such that

$$c(E') \leq 2\left(1 - \frac{1}{n}\right) \sum_{e \in E \setminus F} c_e x'_e = 2\left(1 - \frac{1}{n}\right) \sum_{e \in E \setminus F} c_e x^*_e.$$

Therefore $F \cup E'$ is a k-cut in G and we have that

$$c(F \cup E') = c(F) + c(E') \leq \sum_{e \in F} c_e x^*_e + 2(1 - \frac{1}{n}) \sum_{e \in E \setminus F} c_e x^*_e \leq 2(1 - \frac{1}{n}) \sum_{e \in E} c_e x^*_e.$$

This finishes the proof. Note that the proof also gives a very simple rounding algorithm assuming we have an optimum solution x^* for the LP. Contract all edges with $x^*(e) = 0$, remove all edges e with $x^*(e) = 1$, and use Lemma 5.1 in the residual graph to find the $(k-h)$ smallest degrees vertices.

An LP-based proof of Theorem 2.1: The preceding proof idea also yields a proof of Theorem 2.1, which we sketch here. We are not sure whether this argument has been considered previously. Recall that $\tau(G)$ is the optimum solution value to the tree packing LP, which corresponds to the dual of the k-CUT LP when $k = 2$. When $k = 2$, as we remarked, the LP does not require the upper bound constraints $x(e) \leq 1$ which implies that the dual tree packing LP does not have the z variables. Following Lemma 5.1 we consider a fully fractional optimum solution x^* to the k-CUT LP with $k = 2$ and an optimum dual solution y^* to the dual tree packing LP. We have

$$(n-1) \sum_T y^*_T = (n-1)\tau(G) = \sum_{e \in E} c_e.$$

Consider the partition $\mathcal{P}$ consisting of all the singleton vertices; all edges cross this partition, hence $c(\mathcal{P}) = \sum_e c_e$ and $|\mathcal{P}| = n$. Since $\tau(G) = \frac{\sum_e c_e}{n-1} = \frac{c(\mathcal{P})}{|\mathcal{P}|-1}$ it must be the case that $\frac{\sum_e c_e}{n-1}$ is the network strength which equals the tree packing value. When x^* is not fully fractional we can contract edges with $x^*_e = 0$ and apply the preceding argument. A similar argument can be used to prove the corresponding min-max relation for the fractional packing of bases of a matroid.

6 Characterizing the optimum LP solution

We have seen that the dual of the LP relaxation for k-CUT yields a tree packing that can be used in place of Thorup's recursive tree packing. In this section we show that the two are the same by characterizing the optimum LP solution for a given graph through a recursive partitioning procedure. This yields a nested sequence of partitions of the vertex set of the graph. This sequence is called the principal sequence of

partitions of a graph and is better understood in the more general context of submodular functions [22]. We refer the reader to Fujishige's article for more on this topic [7], and to [5, 17] for algorithmic aspects in the setting of graphs. We also connect the LP relaxation with the Lagrangean relaxation approach for k-CUT considered by Barahona [2] and Ravi and Sinha [24]. Their approach is also built upon the principal sequence of partitions. In order to keep the discussion simple we mainly follow the notation and approach of [24].

Given $G = (V, E)$ and an edge set $A \subseteq E$ let $\kappa(A)$ denote the number of connected components in $G - A$. Recall that the strength of a capacitated graph G, denoted by $\sigma(G)$ is defined as $\min_{A \subseteq E} \frac{c(A)}{\kappa(A)-1}$. The k-CUT problem can be phrased as $\min_{A:\kappa(A)\geq k} c(A)$. However, the constraint that $\kappa(A) \geq k$ is not straightforward. It is, however, not hard to show that $\kappa(A)$ is a supermodular set function over the ground set E. A Lagrangean relaxation approach was considered in [2, 24]. To set this up we define, for any fixed edge set A, a function $g_A : \mathbb{R}_+ \to \mathbb{R}$ as $g_A(b) = c(A) - b(\kappa(A) - 1)$. We then obtain the function $g : \mathbb{R}_+ \to \mathbb{R}$ where $g(b) = \min_{A \subseteq E} c(A) - b(\kappa(A) - 1)$. The quantity $g(b)$ is the attack value of the graph for parameter b and was considered by Cunningham [5] in his algorithm to compute the strength of the graph.

Then, as noted in [2, 24],

$$\min_{A:\kappa(A)\geq k} c(A) \geq \max_{b \geq 0} \min_{A \subseteq E} c(A) + b(k - \kappa(A)) = \max_{b \geq 0} g(b) + b(k-1).$$

Thus $g'(b) = g(b) + b(k-1)$ provides a lower bound on the optimum solution value. [24] describes structural properties of the function g, several of which are explicit or implicit in [5]. We state them below.

- The functions g and g' are continuous, concave and piecewise linear and have no more than $n-1$ breakpoints. The function g is non-increasing in b.
- Under a non-degeneracy assumption on the graph, which is easy to ensure, the following holds. If b is not a breakpoint then there is a unique edge set A such that $g_A(b) = g(b)$. If b is a breakpoint then there are exactly two edge sets A, B such that $g_A(b) = g_B(b)$.
- If b_0 is a breakpoint of g' induced by edge sets A and B with $\kappa(A) > \kappa(B)$ then $B \subseteq A$. In particular $A \setminus B$ is contained in some connected component of $G' = (V, E \setminus B)$.
- Let b_0 be a breakpoint of g' induced by edge set A. Then the next breakpoint is induced by the edge set which is the solution to the strength problem on the smallest strength component of $G' = (V, E \setminus A)$.

The above properties show that the breakpoints induce a sequence of partitions of V which are refinements. Alternatively we consider the sequence of edge sets $A_1, A_2, \ldots,$ obtained by the following algorithm. We will assume that G is connected. Let $A_0 = \emptyset$. Given A_i we obtain $A_{i+1} \supseteq A_i$ as follows. Let $G_i = (V, E \setminus A_{i-1})$. If G_i has no edges we stop. Otherwise let C_{i+1} be the minimum strength connected component of G_i and B_{i+1} be a maximal minimum strength edge set of C_{i+1}. We define $A_{i+1} = A_i \cup B_{i+1}$, and τ_i to be the strength of the component C_{i+1}. That is, $\tau_i = \frac{c(A_i) - c(A_{i-1})}{\kappa(A_i) - \kappa(A_{i-1})}$. The process stops when $A_h = E$. Let $\mathcal{P}_i$ denote the partition of V induced by A_i. Note that $\mathcal{P}_{i+1}$ is obtained from $\mathcal{P}_i$ by replacing C_{i+1} by a minimum strength partition of C_{i+1}, thus $\mathcal{P}_{i+1}$ is a refinement of $\mathcal{P}_i$ and $\mathcal{P}_h$ consists of singleton nodes. Note that Thorup's ideal tree packing is also based on the same recursive decomposition. Let the ith breakpoint of g' to be b_i. It was shown that $b_i = \tau_i$ is precisely the ith breakpoint of the function g' [24].

Ravi and Sinha obtained a 2-approximation for k-CUT as follows. Given the preceding decomposition of G they consider the smallest j such that $|\mathcal{P}_j| \geq k$. If $|\mathcal{P}_j| = k$ they output it and can argue that it is an

optimum solution. Otherwise they do the following. Recall $\mathcal{P}_j$ is obtained from $\mathcal{P}_{j-1}$ by replacing the component C_j in $G - A_{j-1}$ by a minimum strength decomposition of C_j. Let $k' = k - |\mathcal{P}_{j-1}|$. Consider the minimum strength partition of C_j and let $H_1, H_2, \ldots, H_{k'}$ be the connected components of the partition with the smallest shores. Output the cut $A_{j-1} \cup (\cup_{\ell=1}^{k'} \delta(H_\ell))$.

6.1 An optimum LP solution from the decomposition

Given k, as before let j be the smallest index such that $\kappa(A_j) \geq k$. We consider the following solution to the LP:

- $x(e) = 1$ for each $e \in A_{j-1}$.
- $x(e) = \alpha$ for each $e \in A_j \setminus A_{j-1}$, where $\alpha = \frac{k - \kappa(A_{j-1})}{\kappa(A_j) - \kappa(A_{j-1})}$.
- $x(e) = 0$ for each $e \in E \setminus A_j$.

Lemma 6.1 *The solution x is feasible and has objective value*

$$c(A_{j-1}) + \alpha c(B_j) = c(A_{j-1}) + \left(k - \kappa(A_{j-1})\right) b_j.$$

Proof: Let T be any spanning tree. We want to show that $\sum_{e \in T} x(e) \geq k - 1$. For each j, let $\kappa_j = \kappa(A_j)$, and let $\ell_j = |T \cap A_j|$. Then T has ℓ_{j-1} edges of value $x(e) = 1$, and $\ell_j - \ell_{j-1}$ edges of value α. We have

$$\begin{aligned} \sum_{e \in T} x(e) &= \ell_{j-1} + (\ell_j - \ell_{j-1})\alpha \\ &\geq \kappa_{j-1} - 1 + (\kappa_j - \kappa_{j-1})\alpha = k - 1, \end{aligned}$$

where we observe that the RHS of the first line is decreasing in both ℓ_j and ℓ_{j-1}, $\ell_j \geq \kappa_j - 1$, and $\ell_{j-1} \geq \kappa_{j-1} - 1$. To calculate the objective value, we have

$$\sum_{e \in E} x(e) = \sum_{e \in A_{j-1}} c(e) + \sum_{e \in A_j \setminus A_{j-1}} \alpha c(e) = c(A_{j-1}) + \alpha c(B_j)$$

□

The harder part is:

Lemma 6.2 *The solution x attains the optimum value to the LP relaxation.*

Proof: We prove the claim by constructing a dual solution of equal value. See Figure 4.2 for the dual LP.

Recall the definitions of $\mathcal{P}_i, A_i, B_i$, and C_i from above. For each i, let $\kappa_i = \kappa(A_i) = |\mathcal{P}_i|$ be the number of components in the ith partition. The sequence $b_1 < b_2 < \cdots < b_j$ is the breakpoints for g', which is also the strengths of the components $C_1, C_2, \ldots, C_j$. Let Q_i be the partition of C_i corresponding to B_i. An ideal tree packing, following [29], is a convex combination of trees $p : \mathcal{T}(G) \to [0,1]$ s.t. $\sum_T p_T = 1$ with the following properties.

1. For each i, every tree T supported by p induces a tree in the graph $G/\mathcal{P}_i$ obtained by contracting each component of $\mathcal{P}_i$.
2. For each i and each edge $e \in B_i$, p induces $\sum_{T \ni e} p_T = c(e)/b_i$ on e.

Every graph has an ideal tree packing, and (for example) can be constructed recursively as follows. For each C_i, we write each B_i as a sum of b_i (units of fractional) trees in C_i/Q_i (which holds because B_i is a minimum strength cut), and scale it down to a distribution p' of trees in C_i/Q_i with $\sum_{T\ni e} p'_T = c(e)/b_i$ on each edge in B_i. An ideal tree packing now corresponds to the distribution where we take the union of one sampled spanning tree from (the distribution of) each C_i/Q_i.

Let $p : \mathcal{T}(G) \to [0,1]$ be an ideal tree packing. To construct our dual solution, we define nonnegative edge potentials $z(e) \geq 0$ and a tree packing $y(t) \geq 0$ (packing into $c+z$) s.t.

$$y_T = b_j p(T) \qquad \text{for all } T \in \mathcal{T}(G),$$
$$c(e)+z(e) = \begin{cases} \frac{b_j}{b_i} c(e) & \text{for all } e \in B_i \text{ for } i < j \\ c(e) & \text{otherwise.} \end{cases}$$

We first claim that (y,z) is feasible in the dual LP; that is, y is a feasible tree packing w/r/t the augmented capacities $c+z$. Observe that for any edge e, y uses capacity b_j times the capacity by p. We need to show the capacity used by y along any edge e is at most $c(e)+z(e)$. We have two cases.

1. If $e \in B_i$ for some $i < j$, then p uses capacity $\frac{c(e)}{b_i}$. In turn, y uses capacity $\frac{b_j}{b_i} c(e)$. By choice of $z(e)$, we have $c(e)+z(e) = \frac{b_j}{b_i} c(e)$, as desired.

2. If $e \in E \setminus A_{j-1}$, then p uses capacity at most $\frac{c(e)}{b_j}$. In turn, y uses capacity at most $\frac{b_j}{b_i} c(e)$. But $b_i \leq b_j$, so the capacity used by y is $\leq c(e)$.

We now analyze the objective value of our dual solution. We first observe that since each tree supported by y is a tree in $G/\mathcal{P}_j$, we have

$$\begin{aligned}(k-1)\sum_T y_T &= \frac{k-1}{\kappa_j - 1}\sum_T y_T |T \cap A_j| = \frac{k-1}{\kappa_j-1}\sum_{e\in A_j}\sum_{T\ni e} y_T \\ &= \frac{k-1}{\kappa_j-1} b_j \sum_{i\leq j}\frac{1}{b_i}\sum_{e\in B_i} c(e) = \frac{k-1}{\kappa_j-1} b_j \sum_{i\leq j} \kappa_i - \kappa_{i-1} \\ &= \frac{k-1}{\kappa_j-1} b_j (\kappa_j - 1) = (k-1) b_j.\end{aligned}$$

On the other hand, when subtracting out the augmented capacities, we have

$$\begin{aligned}\sum_{e\in E} z(e) &= \sum_{i<j}\sum_{e\in B_i}\left(\frac{b_j}{b_i}-1\right)c(e) = b_j\left(\sum_{i<j}\frac{1}{b_i}\sum_{e\in B_i} c(e)\right) - \sum_{e\in A_{j-1}} c(e) \\ &= b_j \sum_{i<j}(\kappa_i - \kappa_{i-1}) - \sum_{e\in A_{j-1}} c(e) = b_j\left(\kappa_{j-1}-1\right) - \sum_{e\in A_{j-1}} c(e)\end{aligned}$$

Thus the total objective value of our solution, as a function of b_j, is

$$(k-1)\sum_T y_T - \sum_{e\in E} z(e) = \left(k - \kappa_{j-1}\right) b_j + \sum_{e\in A_{j-1}} c(e),$$

as desired. □

Remark One can also verify the optimality of (x,y,z) in the proof above by complimentary slackness conditions. Recall that x and (y,z) satisfy the complimentary slackness conditions if

1. $z_e > 0$ only if $x_e = 1$.
2. $y_T > 0$ only if $\sum_{e \in T} x_e = k-1$.
3. $x_e > 0$ only if $\sum_{T \ni e} y_e = c_e + z_e$.

We address these individually.

1. $z_e > 0$ only if $e \in B_i$ for some $i < j$. In this case, $e \in A_{j-1}$ so $x_e = 1$.
2. If $y_T > 0$ then T is in the support of the ideal tree packing. In particular, T contains exactly $\kappa_j - 1$ edges from A_j and $\kappa_j - 1$ edges from A_{j-1}, so we have

$$\sum_{e \in T} x_e = \sum_{e \in T \cap A_{j-1}} 1 + \sum_{e \in T \cap (A_j \setminus A_{j-1})} \frac{k - \kappa_{j-1}}{\kappa_j - \kappa_{j-1}} = \kappa_{j-1} - 1 + k - \kappa_{j-1} = k - 1,$$

   as desired.
3. If $x_e > 0$, then $e \in B_i$ for some $i \leq j$, so y uses $\frac{b_j}{b_i} c(e) = c(e) + z(e)$ units of capacity of e, as desired.

6.2 Implications of the characterization

We now outline some implications of the preceding characterization of the optimum LP solution.

Ravi and Sinha showed that Lagrangian relaxation lower bound is no weaker than the one provided by LP relaxation. Here we show that they are equivalent.

Theorem 6.3 *The Lagrangian relaxation value is the same as the LP value.*

Proof: Let $\kappa_i = \kappa(A_i) = |\mathcal{P}_i|$ be the number of components after removing A_i. If $\kappa_j = k$ for some j, then the Lagrangian relaxation value is the min k-cut value. Hence it matches the LP value. Otherwise, assume $\kappa_{j-1} < k < \kappa_j$. Since g' is concave, continuous and piecewise linear, one can see that the function g' maximizes at one of the breakpoint, and it is precisely b_j. Indeed, $\kappa_j > k$, so $b_{j+1}(k - \kappa_j) < 0$ is a negative slope. We have

$$g'(b_j) = c(A_{j-1}) - b_j(\kappa_{j-1} - 1) + b_j(k-1) = c(A_{j-1}) + (k - \kappa_{j-1}) b_j.$$

This is precisely the value of the LP in Lemma 6.1. □

The preceding also gives yet another proof that the integrality gap of the LP is $2(1 - 1/n)$.

Second, as we saw, for any value of k, an optimum dual solution to the k-CUT LP can be derived from the ideal tree packing [28, 29]. The last issue is the connection between greedy tree packing and the dual LP. At the high-level it is tempting to conjecture that greedy tree packing is essentially approximating the dual LP via the standard MWU approach. Proving the conjecture formally may require a fair amount of technical work and we leave it for future work. We believe that some insights obtained in [23] could be useful in this context; [23] recasts the LP relaxation for k-CUT into a pure covering LP, and the dual as a pure packing LP that packs forests instead of trees.

References

[1] Ajit Agrawal, Philip Klein, and R Ravi. When trees collide: An approximation algorithm for the generalized Steiner problem on networks. *SIAM Journal on Computing*, 24(3):440–456, 1995.

[2] Francisco Barahona. On the k-cut problem. *Operations Research Letters*, 26(3):99–105, 2000.

[3] Chandra Chekuri, Sudipto Guha, and Joseph Naor. The Steiner k-cut problem. *SIAM Journal on Discrete Mathematics*, 20(1):261–271, 2006.

[4] Chandra Chekuri and Kent Quanrud. Near-linear time approximation schemes for some implicit fractional packing problems. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 801–820. SIAM, 2017.

[5] William H Cunningham. Optimal attack and reinforcement of a network. *Journal of the ACM (JACM)*, 32(3):549–561, 1985.

[6] Rodney G. Downey, Vladimir Estivill-Castro, Michael R. Fellows, Elena Prieto-Rodriguez, and Frances A. Rosamond. Cutting up is hard to do: the parameterized complexity of k-cut and related problems. *Electr. Notes Theor. Comput. Sci.*, 78:209–222, 2003.

[7] Satoru Fujishige. Theory of principal partitions revisited. In *Research Trends in Combinatorial Optimization*, pages 127–162. Springer, 2009.

[8] Harold N. Gabow and K. S. Manu. Packing algorithms for arborescences (and spanning trees) in capacitated graphs. *Mathematical Programming*, 82(1):83–109, Jun 1998.

[9] Michel X Goemans and David P Williamson. A general approximation technique for constrained forest problems. *SIAM Journal on Computing*, 24(2):296–317, 1995.

[10] O. Goldschmidt and D.S. Hochbaum. A polynomial algorithm for the k-cut problem for fixed k. *Mathematics of Operations Research*, pages 24–37, 1994.

[11] Anupam Gupta, Euiwoong Lee, and Jason Li. Faster exact and approximate algorithms for k-cut. In *Proceedings of IEEE FOCS*, 2018.

[12] Monika Henzinger, Satish Rao, and Di Wang. Local flow partitioning for faster edge connectivity. In Philip N. Klein, editor, *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 1919–1938. SIAM, 2017.

[13] David R Karger. *Random Sampling in Graph Optimization Problems*. PhD thesis, Stanford University, February 1995.

[14] David R. Karger. Minimum cuts in near-linear time. *J. ACM*, 47(1):46–76, January 2000.

[15] David R Karger and Clifford Stein. A new approach to the minimum cut problem. *Journal of the ACM (JACM)*, 43(4):601–640, 1996.

[16] Ken-ichi Kawarabayashi and Mikkel Thorup. Deterministic global minimum cut of a simple graph in near-linear time. In Rocco A. Servedio and Ronitt Rubinfeld, editors, *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA, June 14-17, 2015*, pages 665–674. ACM, 2015.

[17] Vladimir Kolmogorov. A faster algorithm for computing the principal sequence of partitions of a graph. *Algorithmica*, 56(4):394–412, 2010.

[18] Pasin Manurangsi. Inapproximability of maximum edge biclique, maximum balanced biclique and minimum k-cut from the small set expansion hypothesis. In *Proc. of ICALP*, volume 80 of *LIPIcs*, pages 79:1–79:14. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2017.

[19] Hiroshi Nagamochi and Toshihide Ibaraki. A linear-time algorithm for finding a sparse k-connected spanning subgraph of a k-connected graph. *Algorithmica*, 7(1-6):583–596, 1992.

[20] Hiroshi Nagamochi and Yoko Kamidoi. Minimum cost subpartitions in graphs. *Information Processing Letters*, 102(2):79 – 84, 2007.

[21] J Naor and Yuval Rabani. Tree packing and approximating k-cuts. In *Proceedings of the twelfth annual ACM-SIAM symposium on Discrete algorithms*, volume 103, page 26. SIAM, 2001.

[22] H. Narayanan. The principal lattice of partitions of a submodular function. *Linear Algebra and its Applications*, 144:179–216, 1991.

[23] Kent Quanrud. Fast and deterministic approximations for k-cut. *CoRR*, abs/1807.07143, 2018.

[24] R Ravi and Amitabh Sinha. Approximating k-cuts using network strength as a lagrangean relaxation. *European Journal of Operational Research*, 186(1):77–90, 2008.

[25] Huzur Saran and Vijay V. Vazirani. Finding k-cuts within twice the optimal. *SIAM J. Comput.*, 24(1):101–108, February 1995.

[26] Alexander Schrijver. *Combinatorial optimization: polyhedra and efficiency*, volume 24. Springer Science & Business Media, 2003.

[27] Mechthild Stoer and Frank Wagner. A simple min-cut algorithm. *Journal of the ACM*, 44(4):585–591, jul 1997.

[28] Mikkel Thorup. Fully-dynamic min-cut. *Combinatorica*, 27(1):91–127, 2007.

[29] Mikkel Thorup. Minimum k-way cuts via deterministic greedy tree packing. In *Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing*, pages 159–166. ACM, 2008.