Trees for Vertex Cuts, Hypergraph Cuts and Minimum Hypergraph Bisection

Harald Räcke
Department of Informatics,
Technische Universität München
raecke@in.tum.de

Roy Schwartz
Computer Science Department,
Technion, Israel.
schwartz@cs.technion.ac.il

Richard Stotz
Department of Informatics,
Technische Universität München
stotz@in.tum.de

ABSTRACT

In the Minimum Hypergraph Bisection problem, the vertex set of a hypergraph has to be partitioned into two parts of equal size so that the number of hyperedges intersecting both parts is minimized. This problem is a natural generalization of the well-studied Minimum Bisection problem in graphs.

In this paper we present a sharp distinction between Minimum Bisection in hypergraphs and graphs. Whereas it is well-known that all bi-criteria approximation algorithms for Minimum Bisection in graphs can be extended to hyergraphs with the exact same guarantees, in this paper we prove that this is not the case when considering true (i.e., non bi-criteria) approximation algorithms. Specifically, we show that Minimum Bisection in Hypergraphs admits an $\tilde{O}(\sqrt{n})$ approximation algorithm (and highlight several special cases where a better approximation ratio is possible). Additionally, we show that the problem is at least as hard as the Densest k-Subgraph problem. Assuming the Dense vs. Random Conjecture [4], no approximation ratio better than $O(n^{1/4-\epsilon})$ is possible. In particular, Minimum Hypergraph Bisection is much harder to approximate than Minimum Bisection in graphs, for which a logarithmic approximation algorithms exist [17].

We also consider the problem of constructing trees that are cut sparsifiers for hypergraph and vertex cuts. While similar trees lie at the heart of powerful algorithms for Minimum Bisection in graphs, we prove that this is not the case for hypergraphs. We give upper and lower bounds to the quality of such trees. Our bounds show that this tree cut sparsifying approach cannot improve the general approximation ratio of Minimum Hypergraph Bisection and Minimum Vertex Bisection.

ACM Reference Format:
Harald Räcke, Roy Schwartz, and Richard Stotz. 2018. Trees for Vertex Cuts, Hypergraph Cuts and Minimum Hypergraph Bisection. In *Proceedings of 30th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA'18).* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/XXXXXX.XXXXXX

1 INTRODUCTION

One major approach for load balancing in large scale scientific computing is to model the problem as a combinatorial graph partitioning problem ([10]). The computation is modeled as a graph, where the nodes represent processes or data and edges represent communication requirements between processes or data objects. The goal is then to partition the graph into roughly equal parts while minimizing the number of edges (the required communication) between different parts. The number of parts should, of course, correspond to the number of processors/cores available on the parallel machine on which the computation should be executed.

Hypergraph partitioning (see e.g. [20]) is an extension of this basic idea where the communication is not modeled by ordinary edges (point-to-point communication) but by hyperedges, instead. This allows to model the communication more accurately but may make solving the underlying partitioning problem more difficult. We refer the reader to the survey [15] (and the references therein) for applications of hypergraph partitioning to the areas of parallel scientific computing, load balancing and congestion minimization.

There exists many different variants of these partitioning problems. As all of these variants are NP-hard they are usually solved by heuristic approaches. From a theoretical perspective many graph partitioning problems are reasonably well understood. Most variants allow either true polylogarithmic approximations (see e.g. [2, 12, 17]) or bi-criteria approximations ([1, 3, 8, 11, 19]). Some of these approaches can also be transferred to hypergraphs. See e.g. [13], which gives polylogarithmic approximation guarantees for *Sparsest Cut*, *Small Set Expansion*, and *ρ-Separator* on hypergraphs.

However, in this paper we show a sharp distinction between Minimum Bisection in hypergraphs and graphs. In this problem the vertex set of a (hyper-)graph has to be partitioned into two parts of equal size so that the number of (hyper-)edges intersecting both parts is minimized. This corresponds to a load balancing problem for two processors.

While the graph version of this problem admits a true $O(\log n)$-approximation [17], and there exist $(O(1), \sqrt{\log n})$ bicriteria approximations[1] for both graphs and hypergraphs, we show that a true approximation with a guarantee of $O(n^{1/4-\epsilon})$ for hypergraphs is unlikely. This hardness is based on the *Hypergraph Dense vs Random Conjecture.* The Dense vs Random Conjecture for ordinary graphs originates from the study of approximation algorithms for the Densest k-subgraph problem. A hypergraph version of this conjecture was introduced by Chlamtáč et al. [5] and was used to

[1]In a bicriteria approximation the algorithm is allowed to return a partition, where the smaller side contains just $\Omega(n)$ vertices instead of exactly $n/2$ vertices. An $(O(1), \sqrt{\log n})$-approximation means that the cost of the (not completely balanced) partition is at most a factor $O(\sqrt{\log n})$ larger than the cost of an optimum bisection.

	Unweighted Vertex	Weighted Vertex	Hypergraph
Cut Tree Quality U.B.	$O(\sqrt{n}\log^{3/4} n)$	$O(\sqrt{nw_{\text{avg}}}\log^{3/4} n)$	$O(\sqrt{nd_{\text{avg}}}\log^{3/4} n)$
Cut Tree Quality L.B.	$\Omega(n^{1/3})$	$\Omega(\sqrt{n})$	$\Omega(\sqrt{n})$
Bisection U.B.			$O(\sqrt{n}\log^{1.25} n)$ $O(h_{\max}\cdot\log n)$
Bisection L.B.	$\Omega(n^{1/8})$	$\Omega(n^{1/8})$	$\Omega(n^{1/4-\varepsilon})$

Table 1: Summary of our results for the Minimum Bisection problem, and the approximation of cuts by trees. Here $h_{\max}$ denotes the maximum size of a hyperedge, and w_{avg} and d_{avg} denote the average weight and degree, respectively, of a vertex in the graph.

prove hardness for the *Minimizing the Union* problem in which one has to select k subsets of vertices such that the size of their union is as small as possible. We also show a direct reduction of the Densest k-subgraph problem to Hypergraph Bisection. This shows that no $n^{\text{poly}(\log\log n)}$ approximation algorithm exists while only assuming ETH, the hypothesis that 3SAT requires exponential time. [14] We also consider the special case when all hyperedges have roughly the same size $\Theta(n^{\alpha})$. For this case we show that the Dense vs Random Conjecture implies a lower bound of $\Omega(n^{\min\{\alpha/2,\alpha(1-\alpha)\}})$ on the approximation ratio.

We complement these lower bounds with an approximation algorithm that obtains an upper bound of $\tilde{O}(\sqrt{n})$ for general hypergraphs, and an upper bound of $\tilde{O}(n^{\min\{\alpha,1-\alpha\}})$ on quasi-uniform hypergraphs with hyperedge size $\Theta(n^{\alpha})$. The largest gap between our upper bound and the lower bound occurs for hyperedges of size $\sqrt{n}$.

In a second part we explore to what extent the techniques that underly many graph partitioning results can be transferred to hypergraphs. The results about Minimum Bisection [17], MinMax Graph Partitioning [19], and Balanced Graph Partitioning [8] are based on a technique that approximates the whole cut-structure of a graph by a tree. Then the respective graph partitioning problem is just solved or approximated on a tree, and one obtains a solution with provable performance guarantees on the original graph. We show that hypergraphs and vertex cuts cannot be represented by a tree-structure with a good approximation guarantee even when allowing vertex cuts in the tree.

In particular we show that this technique cannot obtain a better approximation guarantee than $\Omega(\sqrt{n})$. It is important to note that our lower bounds apply to a *single* tree, as opposed to the more powerful notion of a distribution over trees used for graphs (*e.g.*, see [17]). However, we note that there are previous works that approximate the cut structure of graphs by a tree that are able to obtain a *single* tree that achieves a polylogarithmic guarantee (see [9, 16]). Thus, the comparison between graphs and hypergraphs (as well as vertex cuts) when restricted to a single tree is valid, and our lower bounds (even though restricted to a single tree) exhibit a sharp separation between graphs and hypergraphs. On the positive side we show that one can e.g. find a tree approximation for hypergraph cuts with an approximation guarantee of $\tilde{O}(\sqrt{nd_{\text{avg}}})$, where d_{avg} denotes the average degree of a vertex in the hypergraph. Table 1 gives an overview of our results. Note that the hardness result for vertex bisection will be included in the full version.

1.1 Preliminaries and Definitions

Given a weighted (hyper-)graph $G = (V, E)$ and disjoint sets $A, B \subset V$, an (A, B) *edge cut* is a set of (hyper-)edges whose removal disconnects A and B. We use $\delta_G(A, B)$ to denote the total weight of edges in an (A, B) edge cut of minimum weight. We write $\delta_G(A)$ for $\delta_G(A, V \setminus A)$ and drop the subscript, if the graph is clear from the context.

A *vertex cut* separating two disjoint sets of vertices A and B is a set $X \subseteq V$ so that any path between A and B uses a vertex from X. The vertex cut may include vertices from both A and B. The total weight of the minimum-weight vertex cut between A and B in G is written $\gamma_G(A, B)$. Again, we may drop the subscript if no confusion is possible.

A *vertex separator* (A, B, X) is a set $X \subseteq V$ that separates G into two disconnected pieces A and B. Its *sparsity* is defined by

$$\frac{w(X)}{\min\{w(A), w(B)\} + w(X)} ,$$

where for a vertex set $S \subseteq V$, $w(S)$ describes the total weight of vertices in S. We assume that the weight-function is normalized so that the minimum weight of a vertex is 1.

The vertex separator of minimum sparsity is called *min-ratio cut* of G. Finding the optimal min-ratio cut is NP-complete; Feige et al. [6] give an $O(\sqrt{\log n})$ approximation algorithm for finding the optimal min-ratio cut.

An *edge cut tree* $T = (V_T, E_T)$ of a (hyper)-graph $G = (V, E)$ is a tree with $V \subseteq V_T$; typically, the vertices of G are leaves of the tree. The cut tree is *dominating* if for any two disjoint sets $A, B \subset V$ it holds that $\delta_G(A, B \leq \delta_T(A, B)$, i.e., cuts are larger when measured in the tree. If in addition $\delta_T(A, B) \leq \alpha \cdot \delta_G(A, B)$ holds for all $A, B \subseteq V$ we say that the cut tree has *quality* or *approximation-guarantee* α.

Similarly, a *vertex cut tree* $T = (V_T, E_T)$ of a (hyper)-graph $G = (V, E)$ is a tree with $V \subseteq V_T$. The cut tree is dominating and approximates *vertex cuts* with quality α, if for any two disjoint sets $A, B \subset V$

$$\gamma_G(A, B) \leq \gamma_T(A, B) \leq \alpha \cdot \gamma_G(A, B) .$$

It approximates *edge cuts* with quality α, if for any disjoint sets $A, B \subset V$

$$\delta_G(A, B) \leq \gamma_T(A, B) \leq \alpha \cdot \delta_G(A, B) .$$

2 APPROXIMATING MINIMUM HYPERGRAPH BISECTION

The Minimum Hypergraph Bisection Problem asks to select a set S of $|V|/2$ vertices such that the number of hyperedges leaving S is minimized. We first consider the more general problem of removing k vertices from a hypergraph and reduce it to the case of simple graphs, highlighting several special cases. Then we use this procedure in an $\tilde{O}(\sqrt{n})$ approximation algorithm for Minimum Hypergraph Bisection.

2.1 Cutting k Vertices from a Hypergraph

This section shows how to efficiently remove k vertices from a hypergraph so that the number of hyperedges in the cut is minimized. This problem is sometimes called *Unbalanced k-cut.*

Given the hypergraph H, replace any hyperedge h with a clique on its vertices. Let the weight of each edge in the clique be $1/(|h|-1)$ and call the resulting graph G'. The cut cost of H and G' are related via the following inequality, where $h_{\max}$ is the size of the largest hyperedge.

Lemma 1. *For any set S of k vertices, we have $\delta_H(S) \le \delta_G(S) \le \min\{k, h_{\max}/2\}\delta_H(S)$.*

Proof. For any hyperedge h in the cut in H, there are at least $|h|-1$ weighted edges in the cut in G. As their edge weight is $1/(|h|-1)$, the first inequality follows.

The number of edges that cross the cut S in G' for a hyperedge h of G is $|h \cap S| \cdot |h \cup S|$. For $h \ge 2k$ this is at most $k(|h|-k)$. Otherwise, it is at most $\lfloor |h|/2 \rfloor \cdot \lceil |h|/2 \rceil \le |h|^2/4$. In the first case the cut increases by a factor of at most $k(|h|-k)/(|h|-1) \le k$, and in the second case by at most $|h|^2/(4(|h|-1)) \le |h|/2$ since $|h| \ge 2$. □

Using a convex combination of decomposition trees the optimal unbalanced k-cut in a graph can be approximated up to a factor of $O(\log n)$ [17]. Applying this procedure to the graph G' results in an approximation algorithm for finding an unbalanced k-cut in a hypergraph.

Proposition 1. *There exists a $\min\{k, h_{\max}/2\} \cdot O(\log n)$ approximation algorithm for finding the unbalanced k-cut in a hypergraph, where $h_{\max}$ is the size of the largest hyperedge.*

If all hyperedges have size at least k, no hyperedge can be completely contained in a set of k vertices. In this case, the unbalanced k-cut problem becomes the Minimizing k-Union problem, studied recently by Chlamtáč et al. [5]. They obtain the following result, where the $\tilde{O}$-notation hides polylogarithmic factors.

Proposition 2 ([5]). *If all hyperedges have at least size k, and $k = n^{1-\alpha}$, then there is an $\tilde{O}(n^{\alpha(1-\alpha)+\varepsilon})$ approximation algorithm for the unbalanced k-cut problem, for every sufficiently small $\varepsilon > 0$.*

2.2 Approximation Algorithm for Minimum Hypergraph Bisection

In this section, we adapt an algorithm for graph bisection by Feige et al. [7] to prove the following theorem.

Theorem 1. *There is a polynomial-time $O(\sqrt{n}\log^{5/4} n)$ approximation algorithm for Minimum Hypergraph Bisection.*

Proof. The algorithm first guesses the value OPT of an optimum solution. It then uses an approximation algorithm for Sparsest Cut and recursively cuts pieces from the graph until it does not find a cut of sparsity $\alpha\,\mathrm{OPT}/k$ anymore (in any of the pieces). Here, α denotes the approximation guarantee of the algorithm used (e.g. $\alpha = O(\sqrt{\log n})$ [13]), and k is a parameter to be determined later. Note that this implies that no cut of sparsity below OPT/k can exist within a piece.

Lemma 2. *The total weight of edges cut in the first phase is at most $\alpha n \log n \cdot \mathrm{OPT}/k$.*

Proof. Every cut S_i of the first phase fulfills the inequality $|\delta(S_i)|/|S_i| \le \alpha\,\mathrm{OPT}/k$, where S_i denotes the smaller side of the cut. This gives $|\delta(S_i)| \le \alpha|S_i|\,\mathrm{OPT}/k$.

We can amortize the increase $\delta(S_i)$ of the total cut-cost in the i-th step to the vertices on the smaller side. Since during the whole construction a vertex appears on the smaller side of a cut at most $\log n$ times we amortize at most $\alpha \log n \cdot \mathrm{OPT}/k$ against a single vertex. Summing over all vertices gives the lemma. □

Let G_i denote the sub-graphs/pieces returned after the first phase. We fix some optimal bisection represented as a black and white coloring of the vertices. We say a subgraph G_i has *minority color* white, if it contains fewer white vertices than black vertices; otherwiese, its minority color is black. The vertices that have the minority color are called *minority vertices.* The following lemma gives a bound on the number of minority vertices at the end of the first phase.

Lemma 3. *There exist less than k minority vertices by the end of the first phase.*

Proof. Let r_i denote the number of minority vertices in part G_i and let OPT_i be the number of hyperedges of the (fixed) optimal bisection inside G_i. Note that $\sum_i \mathrm{OPT}_i \le \mathrm{OPT}$, as the graphs G_i are disjoint. As no cut of sparsity below OPT/k exists, every i must fulfill

$$\frac{\mathrm{OPT}_i}{r_i} \ge \frac{\mathrm{OPT}}{k} \quad .$$

Summing over all i gives

$$\frac{\sum_i \mathrm{OPT}_i}{\mathrm{OPT}} \ge \frac{\sum_i r_i}{k} \quad .$$

As the left-hand side is at most 1, the same holds for the right-hand-side. The total number of minority vertices $\sum_i r_i$ is therefore at most k. □

The second phase of the algorithm aims to approximate the optimum cut within each piece G_i. For this we "guess" in each subgraph the number of minority vertices k_i and remove so many vertices using the algorithm from Proposition 1. These guesses are made one-by-one within a dynamic program as described in [7]. We can amortize the cost for this step against the cost OPT_i that the optimum solution has within G_i.

For guesses k_i we pay at most

$$O(\log n)\sum_i k_i\,\mathrm{OPT}_i \le O(\log n)k\,\mathrm{OPT} \qquad (1)$$

in this step. A simple dynamic programming approach see e.g. [7] allows to find the optimum "guesses" k_i.

We choose $k = \sqrt{\alpha n}$ to balance the approximation factors in Equation 1 and Lemma 2. This gives an overall approximation guarantee of $O(\log n\sqrt{\alpha n}) = O(\log^{5/4} n\sqrt{n})$ as desired. □

If either all hyperedges are large or all hyperedges are small we can obtain the following improved bounds.

Theorem 2. *If all hyperedges have size at least $\Omega(n^\alpha)$, there exists an $\tilde{O}(n^{1-\alpha})$ approximation algorithm for Minimum Hypergraph Bisection.*

If all hyperedges have size at most $O(n^\alpha)$, there is an $\tilde{O}(n^\alpha)$-approximation algorithm for Minimum Hypergraph Bisection.

Proof. If all hyperedges have at least size $z = \Omega(n^\alpha)$, we choose $k = z$ for the first phase of the algorithm. Then the total number of minority vertices for the second phase is only z. This means we can use the algorithm from Proposition 2 for the second phase. The approximation factor of the overall algorithm is then $\tilde{O}(n^{1-\alpha}) + \tilde{O}(n^{\alpha(1-\alpha)+\varepsilon}) \in O(n^{1-\alpha})$.

If all hyperedges have size *at most* $O(n^\alpha)$, applying Lemma 1 and a logarithmic approximation algorithm for Minimum Bisection in simple graphs gives an $\tilde{O}(n^\alpha)$ approximation algorithm for Minimum Hypergraph Bisection. □

2.3 Hardness Results

In this section, we prove that the existence of an efficient algorithm for Minimum Hypergraph Bisection with approximation ratio $n^{\mathrm{poly}(\log\log n)}$ would violate the Exponential Time Hypothesis (ETH). Under the stronger *Hypergraph Dense vs Random Conjecture*, we show that the approximation ratio of Minimum Hypergraph Bisection cannot be better than $O(n^{1/4-\varepsilon})$.

The Dense vs Random Conjecture originates from the study of strong approximation algorithms for the Densest k-Subgraph problem. Let the *log-density* of a (hyper-)graph on n vertices be $\log_n(d_{\mathrm{avg}})$, where d_{avg} denotes the average vertex degree. For hypergraphs, the conjecture asks to distinguish a random r-uniform hypergraph $G = \mathcal{G}_{n,p,r}$ with $p = n^{1+\alpha-r}$ (so G has log-density α) and an adversarially chosen hypergraph G' containing a subhypergraph with log-density β on k vertices.

Conjecture 1 ([5]). *For all constant r and $0 < \beta < r-1$, for all sufficiently small $\varepsilon > 0$, and for all k such that $k^{1+\beta} \le n^{(1+\alpha)/2}$, we cannot solve Hypergraph Dense vs Random with log-density α and planted log-density β in polynomial time (w.h.p) when $\beta < \alpha - \varepsilon$.*

Chlamtáč et al. apply the conjecture to show that *Minimizing k-Union* (MkU) does not admit an $O(m^{1/4-\varepsilon})$ approximation algorithm. In this problem, we are given a hypergraph $G = (V, E)$ of n vertices and m hyperedges, and an integer $k \le m$. The goal is to select k hyperedges so that their union has minimal size. We extend Chlamtáč et al.'s argument by parameterizing it with the vertex degree of the hypergraph G. Later, MkU serves as the starting point for proving the hardness of Minimum Hypergraph Bisection.

We say that an instance of MkU is *quasi α-uniform*, if the degree of all vertices is $\Theta(n^\alpha)$ and all hyperedges have constant size r. We denote a quasi α-uniform instance of MkU with (G, n, m, α, r). The following lemma shows a hardness for quasi α-uniform instances of MkU.

Lemma 4. *Assuming Conjecture 1, there is no polynomial-time $O(m^{\min\{\alpha/(2+2\alpha),\alpha/(1+\alpha)^2\}-\varepsilon})$ approximation algorithm for quasi α-uniform Minimizing the Union.*

Proof. In order to establish the lemma we first show some facts about $\mathcal{G}_{n,p,r}$. The proof of these facts has been deferred to the full version.

Claim 1. *A random graph $\mathcal{G}_{n,p,r}$ fulfills the following properties.*

1. *With high probablity $\mathcal{G}_{n,p,r}$ with $p = n^{1+\alpha-r}$ has vertex degree $\Theta(n^\alpha)$.*
2. *Any set of n edges in a $\mathcal{G}_{n,p,r}$ with $p = n^{1+\alpha-r}$ covers at least $\frac{1}{2e}(n/p)^{1/r}$ vertices, with high probability.*
3. *Any set of $n^{(1+\alpha)/2}/r$ edges in a $\mathcal{G}_{n,p,r}$ with $p = n^{1+\alpha-r}$ covers at least $n^{(1+\alpha)/2-\varepsilon}/e$ vertices, with high probability, if $\alpha < 1$, r is sufficiently large and ε is a small constant.*

The planted subhypergraph of the adversary contains $k^{1+\beta}/r$ many hyperedges as $k^{1+\beta}$ is the sum of all degrees in the induces graph and r is the size of a hyperedge. We choose $\ell = k^{1+\beta}/r$. It follows that in the planted instance the size of a minimum ℓ union is at most k. Note that the above choice for ℓ does not fix ℓ but just the relationship between ℓ and k because k is a free parameter of the Dense vs. Random instance.

In the following we show that w.h.p. the number of vertices contained in the union of ℓ hyperedges in a random instance is considerably larger than k. As it is hard to distuingish between random instances and a planted instace the lower bound follows.

Let $\bar{k}$ denote the size of the minimum ℓ union in a random instance. We show that $\bar{k}$ is large with high probability. We distinguish two cases according to the value of α.

$\boldsymbol{\alpha > 1}$. We choose $\ell = n$ (note that this in fact chooses k for the Dense vs. Random instance, because we have set $\ell = k^{1+\beta}/r$ before). We first show that $k^{1+\beta} \le n^{(1+\alpha)}$ as this is a constraint on k that must be fulfilled for Conjecture 1. Indeed, $k^{1+\beta} = r\ell = rn < n^{(1+\alpha)/2}$ for $\alpha > 1$.
Fact 2 implies that with high probability $\bar{k}$ will be at least $\Theta(n/p)^{1/r}$. Let m denote the total number of edges in the random instance. Fact 1 implies that, with high probability, $m = \Theta(n^{1+\alpha})$. We use this fact and rewrite $(n/p)^{1/r}$ with respect to m:

$$\begin{aligned}(n/p)^{1/r} &= \Theta\left(\left(m^{1/(1+\alpha)}n^{r-(1+\alpha)}\right)^{1/r}\right)\\ &= \Theta\left(\left(m^{1/(1+\alpha)}m^{r/(1+\alpha)-1}\right)^{1/r}\right)\\ &= \Theta\left(m^{1/(r(1+\alpha))-1/r+1/(1+\alpha)}\right)\quad .\end{aligned}$$

In the first equality, we used that $p = n^{1+\alpha-r}$. As $k = \Theta(n^{1/(1+\beta)}) = \Theta(m^{1/(1+\alpha)(1+\beta)})$ (with r being constant), it follows that

$$\begin{aligned}\bar{k}/k &= \Theta\left(m^{\frac{1}{r(1+\alpha)}-\frac{1}{r}+\frac{1}{1+\alpha}-\frac{1}{(1+\alpha)(1+\beta)}}\right)\\ &> \Theta\left(m^{-\frac{1}{r}+\frac{1}{1+\alpha}-\frac{1}{(1+\alpha)^2}-\varepsilon/4}\right)\\ &= \Theta\left(m^{\alpha/(1+\alpha)^2-\varepsilon}\right) ,\end{aligned}$$

using $\beta = \alpha - \varepsilon$ and $1/r < \varepsilon/2$ sufficiently small.

$\alpha \le 1$. We choose $\ell = n^{(1+\alpha)/2}/r$. This choice is valid as $k^{1+\beta} = \ell r \le n^{(1+\alpha)/2}$. Fact 3 implies that with high probability $\bar{k}$ will be at least $\Theta(n^{(1+\alpha)/2-\varepsilon})$, if r is sufficiently large. Again $m = \Theta(n^{1+\alpha})$ by Fact 1, so

$$\bar{k} = \Theta(m^{1/2-\varepsilon/(1+\alpha)}) .$$

On the other hand, $k = \Theta(m^{1/(2(1+\beta))})$, so the gap $\bar{k}/k$ is

$$\begin{aligned}\bar{k}/k &> \Theta(m^{1/2-\varepsilon/(1+\alpha)}) \cdot \Theta(m^{-1/(2(1+\beta))})\\ &= \Theta(m^{\beta/(2(1+\beta))-\varepsilon/(1+\alpha)})\\ &\ge \Theta(m^{\alpha/(2(1+\alpha))-\varepsilon'}) ,\end{aligned}$$

using $\beta = \alpha - \varepsilon$ and ε' sufficiently small.

The combination of both cases directly yields the lemma. □

Theorem 3. *An $f(|V|)$-approximation algorithm for the Minimum Hypergraph Bisection problem on hypergraphs $G = (V, H)$ implies an $O(f(|H'|))$-approximation algorithm for Minimizing k-Union on hypergraphs $G' = (V', H')$.*

Proof. Given an instance of MkU $G' = (V', H')$ with $n = |V'|$ and $m = |H'|$, the reduction constructs the following hypergraph $G = (V, H)$. Essentially, we switch hyperedges and vertices, adding a supervertex incident to every new hyperedge. Formally, the vertex set V consists of a supervertex w and a vertex v_i for each hyperedge $h'_i \in H'$. For $v'_j \in V'$, a hyperedge h_j connects w with all vertices v_i fulfilling $v'_j \in h'_i$. Additionally, there exist p *padding vertices* u_ℓ, with $p = \max\{m+1-2k, 2k-m-1\}$, where k is the number of hyperedges that must be chosen in the MkU instance. If $k > (m + 1)/2$, the padding vertices are connected with the supervertex w by infinite-cost edges. Otherwise, they are not connected to any vertex.

As every hyperedge in the graph is incident to the supervertex w, any bisection (V_1, V_2) of G must have one side without any internal hyperedges (namely the side not containing w). Let V_1 be this side. Furthermore, let $\bar{n} = |V| = m + 1 + p$ denote the number of vertices in G. Note that $\bar{n} \in \Theta(m)$.

First, suppose that k is smaller than $(m + 1)/2$, thus $p = m + 1 - 2k$. Let (V_1, V_2) be the bisection of G obtained by an $f(|V|)$-approximation algorithm. Let b be the cost of the bisection (V_1, V_2). Both V_1 and V_2 consist of $\bar{n}/2 = m + 1 - k$ vertices. Therefore both of them contain at least k non-padding vertices. These non-padding vertices correspond to sets in the MkU instance. As V_1 has no internal hyperedges, the hyperedges incident to vertices of V_1 are exactly the ones cut by the bisection. Therefore the corresponding sets of the MkU instance cover b items of the ground set.

Next, observe that the k *optimal* sets for the MkU instance, whose union has cardinality $\mathrm{OPT}_{\mathrm{MkU}}$, correspond to k vertices in the graph G. Together with the p padding vertices, they form a bisection of G with cost $\mathrm{OPT}_{\mathrm{MkU}}$. It follows that $b \le f(|V|)\,\mathrm{OPT}_{\mathrm{MkU}}$, i.e., the algorithm has constructed an $f(|V|)$-approximation for the MkU instance. As $|V| = \Theta(|H'|)$, this approximation is also an $O(f(|H'|))$ approximation.

Now suppose that $k > (m + 1)/2$ and therefore $p = 2k - m - 1$. We use a similar argument as above. Observe that an $f(|V|)$-approximate bisection (V_1, V_2) has both V_1 and V_2 of size k. Recall that in this case the padding vertices are connected with the supervertex w by infinite-cost edges. Therefore, with out loss of generality, V_1 does not contain any padding vertices. Let b denote the cost of the bisection (V_1, V_2). The vertices of V_1 correspond to k sets in the MkU instance, covering b items from the ground set. An optimal solution of the MkU instance with cost $\mathrm{OPT}_{\mathrm{MkU}}$, however, corresponds to a solution to the Hypergraph Bisection instance with cost at most $\mathrm{OPT}_{\mathrm{MkU}}$. It follows as above that $b \le f(|V|)\,\mathrm{OPT}_{\mathrm{MkU}}$, i.e., the algorithm has constructed an $O(f(|E'|))$-approximation for the MkU instance. □

Corollary 1. *Assuming the Dense vs Random Hypothesis, there is no $O(n^{1/4-\varepsilon})$ approximation algorithm for Minimum Hypergraph Bisection, where n denotes the number of vertices of the Hypergraph.*

Furthermore, there is no $O(n^{\min\{\alpha/2, \alpha(1-\alpha)\}-\varepsilon})$ approximation algorithm for Minimum Hypergraph Bisection on hypergraphs with hyperedge size $\Theta(n^\alpha)$.

Proof. The reduction performed in the proof of Theorem 3 transforms a quasi α-uniform instance (G', n, m, α, r) of MkU into an instance $G = (V, E)$ of Minimum Hypergraph Bisection. The graph G has $\bar{n} = \Theta(m)$ vertices and $\bar{m} = n$ hyperedges. As the instance of MkU is quasi α-uniform, every vertex of G' has degree $\Theta(n^\alpha)$, and, given that the size r of the hyperedges is constant, $m = \Theta(n^{1+\alpha}/r) = \Theta(n^{1+\alpha})$. The hyperedges of G therefore have size $\Theta(n^\alpha) = \Theta(m^{\alpha/(1+\alpha)}) = \Theta(\bar{n}^{\alpha/(1+\alpha)})$. With $\gamma = \alpha/(1+\alpha)$, the hyperedges of G thus all have size $\Theta(\bar{n}^\gamma)$.

Lemma 4 implies that quasi α-uniform instances of MkU cannot be approximated up to factor $m^{\min\{\alpha/(2+2\alpha), \alpha/(1+\alpha)^2\}-\varepsilon}$. As

$$\frac{\alpha}{(1+\alpha)^2} = \frac{\alpha}{1+\alpha} \cdot \frac{1}{1+\alpha} = \frac{\alpha}{1+\alpha} \cdot \left(1 - \frac{\alpha}{1+\alpha}\right) ,$$

it follows that Minimum Hypergraph Bisection with hyperedge sizes $\Theta(n^\gamma)$ cannot be approximated with factor $\bar{n}^{\min\{\gamma/2, \gamma(1-\gamma)\}}$. □

Using a similar reduction, we show that an f-approximation algorithm for the Minimum Hypergraph Bisection problem implies an f^2-approximation algorithm for Densest k-Subgraph. The Densest k-Subgraph is notoriously hard and has recently been shown not to admit approximation algorithms of approximation ratio $n^{\mathrm{poly}(\log\log n)}$ under the Exponential Time Hypothesis [14]. Under the stronger GAP-ETH hypothesis, no subpolynomial approximation ratio is possible.

Theorem 4. *An f-approximation algorithm for Minimum Hypergraph Bisection implies an f^2-approximation algorithm for Densest k-Subgraph.*

Proof. The decision version of Densest k-Subgraph asks to return a set of k vertices such that the induced subgraph contains L edges. An f-approximation algorithm for a (k, L) instance returns a set of k vertices whose induced subgraph contains L/f edges, if a subgraph inducing L edges exists.

Given an instance $G = (V, E)$ of Densest k-Subgraph, guess the number of edges L in the optimal subgraph. Consider the problem of choosing L edges, minimizing the number of edges incident to it. This is an instance of Minimizing k-Union.

Following Theorem 3, an f-approximation algorithm for Minimum Hypergraph Bisection is an f-approximation to the Minimizing k-Union problem. Its solution translates into a subset $S \subseteq V$ with $|S| \leq f \cdot k$ and $|E_G(S)| \geq L$.

A random subset S' of S with k vertices induces, in expectation, at least L/f^2 edges. This is an f^2-approximation to the optimal value L. Using the method of conditional expectations, the random selection can be derandomized. □

Corollary 2. *Assuming the Exponential Time Hypothesis, there is no efficient approximation algorithm for Minimum Hypergraph Bisection with approximation ratio* $n^{poly(\log\log n)}$.

Proof. Follows from Theorem 4 and [14]. □

3 CUT TREES FOR VERTEX AND HYPERGRAPH CUTS

In this section we explore a more general approach to solving cut problems in hypergraphs. We try to approximate the cut-structure of a hypergraph $G = (V, E)$ by either an edge cut tree or a vertex cut tree. For ordinary graphs this type of approach lies at the heart of the $O(\log n)$-approximation algorithm for minimum bisection.[2]

3.1 Constructing vertex cut trees with quality $\tilde{O}(\sqrt{W})$

This section demonstrates the construction of a vertex cut tree of quality $\tilde{O}(\sqrt{W})$, for weighted graphs of total weight W. Recall that the minimum weight of a vertex is 1.

Construction. The construction of the tree uses an algorithm $\mathcal{A}$ returning an α-approximate min-ratio vertex cut. Recall that a polynomial-time algorithm with $\alpha \in O(\sqrt{\log n})$ exists [6].

Use algorithm $\mathcal{A}$ to find the min-ratio vertex cut (A, B, X) in the graph. Let $G_1^X, G_2^X, \ldots$ be the connected components after removing the separator X. Repeat the process on each subgraph until no cut of sparsity less than $\alpha f(W)$ can be found, for some function f to be chosen later. This implies that no cut of sparsity less than $f(W)$ exists.

Let $G_1, G_2, \ldots$ denote the subgraphs remaining at the end of this process with $G_i = (V_i, E_i)$ and let $\mathcal{S}$ denote the union of all separators found. The cut tree is constructed by first adding a root vertex r of weight $w(\mathcal{S})$. For any vertex $s \in \mathcal{S}$, the root obtains a child of weight $w(s)$. For any subgraph G_i, add a child t_{G_i} of infinite weight to the root. Any vertex v in V_i is added as a leaf of weight $w(v)$ to t_{G_i}. A sketch of the construction can be seen in Figure 1.

[2]The result by Räcke [17] that gives an $O(\log n)$-approximation for minimum bisection uses an approximation of the graph by a *convex combination* of cut trees. Here we try to approximate the cut-structure by a single tree.

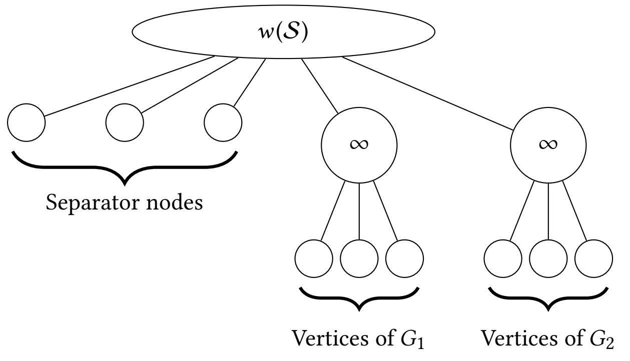

Figure 1: Construction of the vertex cut tree.

Analysis. The following lemma shows that the tree constructed according to the above algorithm is dominating.

Lemma 5. *For any two disjoint sets* $A, B \subseteq V$ *we have* $\gamma_G(A, B) \leq \gamma_T(A, B)$.

Proof. Let X be some minimum vertex cut separating A and B in the tree T.

Any non-infinite-weight vertex cut in T may only contain leaves and the root. By associating the root with the vertex set $\mathcal{S}$, the minimum cut X corresponds to a set $X' \subset V$ of vertices of the graph G. Note that X and X' have identical weight. We show that X' separates any pair of vertices $u \in A$ and $v \in B$ in the graph G.

The claim is obvious if either $a \in X$ or $b \in X$; suppose $a, b \notin X$. This implies that neither a nor b belong to the set $\mathcal{S}$, defining the root. If a and b belong to the same subgraph G_i, then they are not separated by X in the tree, which is a contradiction. If a and b belong to different subtrees G_i and G_j, then they can only be separated in the tree, if the root belongs to the cut X. This implies that the vertices of $\mathcal{S}$ belong to X'. As the vertices of $\mathcal{S}$ separate all subgraphs G_i by construction, it follows that a and b are separated by X'. □

Lemma 6. *If G is connected, we have* $\gamma_T(A, B) \leq O(\sqrt{\alpha \log n\, W}) \cdot \gamma_G(A, B)$.

Proof. In the following we construct a vertex cut X separating A and B in T. First, we include the root r_T of T into this cut. We derive a bound on the weight $w(r_T)$ of the root as follows. The weight $w(r_T)$ is the weight of all vertices in the separator $\mathcal{S}$. For any separator S_i found during the construction we have

$$\frac{w(S_i)}{w(A_i) + w(S_i)} \leq \alpha f(W) \ ,$$

where A_i denotes the side of the cut that has the smaller number of vertices (not necessarily the smaller weight). This gives

$$w(S_i) \leq \Big(1 - \alpha f(W)\Big)^{-1} \alpha f(W) \cdot w(A_i) \leq 2\alpha f(W) \cdot w(A_i) \ ,$$

for sufficiently large n if we choose f such that $\alpha f(W) = o(1)$. We can amortize the weight $w(S_i)$ of the separator against the weight $w(A_i)$ of vertices in A_i. Since a vertex can be on the smaller side of the cut at most $\log n$ times we get that

$$\begin{aligned} w(\mathcal{S}) &= \sum_i w(S_i) \le 2\alpha \log n\, f(W) \sum_i w(A_i) \\ &\le 2\alpha \log n\, f(W) W \ . \end{aligned}$$

Since the cost of the optimum cut is at least one this gives

$$w(r_T) \le 2\alpha \log n\, f(W) W \cdot \gamma_G(A,B) \ . \qquad (2)$$

In addition to the root r_T we add some leaf vertices from every sub-graph $G_i = (V_i, E_i)$ to the cut X. Let $A_i = A \cap V_i$ and $B_i = B \cap V_i$ (note that these sets might be empty). If $w(A_i) \le w(B_i)$ we add the leaf vertices corresponding to A_i; otherwise we add the vertices corresponding to B_i. This forms an (A,B)-cut: a vertex $a \in A_i$ is separated from B-vertices of other sub-graphs G_i, because of the root r_T, and it is separated from B-vertices of its own sub-graph G_i because either A_i or B_i is in the cut. We use X_i to denote the set of vertices that we added in this step for graph G_i.

Let X_i^* denote the intersection of the optimal (A,B)-cut with the sub-graph G_i. Since the sparsity of any cut in G_i is at least $f(W)$ we get that

$$\frac{w(X_i^*)}{w(X_i \cup X_i^*)} \ge f(W) \ ,$$

because X_i^* is a cut that separates X_i (i.e., either A_i or B_i). Consequently, $w(X_i) \le w(X_i \cup X_i^*) \le f(W)^{-1} w(X_i^*)$. Summing this over all i gives

$$\sum_i w(X_i) \le f(W)^{-1} \sum_i w(X_i^*) \le f(W)^{-1} \gamma_G(A,B) \ . \quad (3)$$

Combining equations 2 and 3 gives that the cut X that we constructed fulfills

$$w(X) \le (f(W)^{-1} + 2\alpha \log n f(W) W) \cdot \gamma_G(A,B) \ .$$

Choosing $f(W) = 1/\sqrt{\alpha \log n\, W}$ gives the lemma. □

Combining Lemma 5 and Lemma 6 gives the following theorem.

Theorem 5. *For a graph G with total vertex weight W, we can construct an $O(\log^{3/4} n)\sqrt{W}$-approximate vertex cut tree in polynomial time.*

Application to Hypergraph Bisection. It is possible to also apply Theorem 5 for approximating hypergraph cuts. In order to do this we construct a vertex weighted graph G' from a given hypergraph G, such that vertex cuts in G' correspond to hyperedge cuts in G.

Let $G = (V, H)$ be the given hypergraph. We construct a bipartite graph $G' = (V \cup H, E, w)$, as follows. The vertex set V' of G' consists of both the vertices and the hyperedges of G. Any vertex $v \in V$ is connected in G' to the hyperedges that it is incident to in G by a simple edge. The weight $w(v)$ is set to $w(v) = \deg_G(v) + 1$, where $\deg_G(v)$ is the degree of v in G. The weight of a vertex in H is set to 1.

The following claim shows that edge cuts in G correspond to vertex cuts in G'.

Lemma 7. *For any two sets $A, B \subseteq V$: $\gamma_{G'}(A,B) = \delta_G(A,B)$.*

Proof. An edge cut in G between A and B induces a vertex cut in G' that also separates A and B. This gives $\gamma_{G'}(A,B) \le \delta_G(A,B)$. On the other hand, a minimum (A,B) vertex cut in G' will not use a vertex $v \in V$ as it would be cheaper to use all hyperedges connected to v instead. Hence, it only consist of hyperedges, which give rise to an (A,B) edge cut in G, and therefore $\gamma_{G'}(A,B) \ge \delta_G(A,B)$ holds. □

Corollary 3. *There is a $\sqrt{d_{\text{avg}}}$ approximation algorithm for minimum bisection in hypergraphs, where d_{avg} is the average vertex degree.*

Proof. We first construct a vertex cut tree T for G'. Then we use dynamic programming to compute the minimum vertex cut in T such that one half of the vertices of V are separated from the other half. This induces a bisection in G'. Because of the approximation guarantee of the cut tree this is an $O(\log^{3/4} n \cdot \sqrt{nd_{\text{avg}}})$-approximation. The full version contains the details of the dynamic programming approach. □

3.2 Lower bounds

In this section we show various lower bounds for approximating vertex cuts or hypergraph cuts by cut trees. We first show that in order to get any reasonable approximation guarantee one needs to consider vertex cut trees instead of edge cut trees. This is in strong contrast to ordinary graphs where edge cut trees already give a polylogarithmic approximation guarantee [18].

3.2.1 *Edge Cut Trees for Hyperedge Cuts.*

Theorem 6. *There exists a hypergraph on n vertices such that the quality of any edge cut tree is $\Omega(n)$.*

Proof. Consider a hypergraph $G = (V, H)$ with a single hyperedge e spanning all n nodes of the hypergraph. Let T be a dominating edge cut tree of G. Each edge in T defines a cut in the hypergraph of weight 1, thus without loss of generality all edges in T have weight 1.

Consider two different vertex sets $\emptyset \ne S_1, S_2 \subset V$ with $|S_1|, |S_2| < n/2$ and let $\bar{X} = V \setminus X$ for any vertex set X. Let F_1, F_2 be minimum-size sets of tree edges separating S_1 from $\bar{S_1}$ and S_2 from $\bar{S_2}$, respectively. We show that F_1 and F_2 cannot be equal.

Assume for a contradiction that $F_1 = F_2 = F$. Removing F from the tree decomposes it into clusters. The vertices of V in a cluster classify it in one of four groups: $S_1 \cap S_2$, $S_1 \cap \bar{S_2}$, $\bar{S_1} \cap S_2$ and $\bar{S_1} \cap \bar{S_2}$. We say that that a cluster is an (S_1, S_2)-cluster, if all of its vertices are from $S_1 \cap S_2$; the nomenclature for the other groups is analogous.

We observe that if F contains an edge e linking an (S_1, S_2)-cluster to an $(S_1, \bar{S_2})$-cluster, then $F \setminus \{e\}$ still disconnects S_1 and $\bar{S_1}$, which is a contradiction to the minimality of F. Similarly, no edge $e \in F$ can connect an (S_1, S_2)-cluster to an $(\bar{S_1}, S_2)$-cluster, an $(\bar{S_1}, \bar{S_2})$-cluster to an $(\bar{S_1}, S_2)$-cluster, and an $(\bar{S_1}, \bar{S_2})$-cluster to an $(S_1, \bar{S_2})$-cluster.

Define the disjoint sets $A = (S_1 \cap S_2) \cup (\bar{S_1} \cap \bar{S_2})$ and $B = (S_1 \cap \bar{S_2}) \cup (\bar{S_1} \cap S_2)$. Sets A and B cannot be empty as $|S_1|, |S_2| < n/2$ and both are nonempty; also $A = \bar{B}$. We show that A and B are disconnected in G, which is a contradiction to fact that G is connected.

On the one hand, A and B must be separated in the tree by removing F. To see this, observe that the removal of F disconnects any $x \in S_1 \cap S_2$ must be disconnected from all vertices in $\bar{S_1} \cup \bar{S_2} \supset B$. Similarly, the removal of F disconnects $x \in \bar{S_1} \cap \bar{S_2}$ from all vertices in $S_1 \cup S_2 \supset B$. On the other hand, the clusters containing the vertices of A and the clusters containing the vertices of B are not connected through edges of F, as shown above. It follows that

$F \setminus F = \emptyset$ disconnects A and B in the tree. This means that A and B are disconnected in G, which is a contradiction.

The total number of tree edges can be bounded without loss of generality by $2n$. To see this, note that if a leaf of the tree does not belong to V, the edge leading to it is never removed by a minimum cut separating vertex sets of V. Furthermore, if any tree vertex not belonging to V has only one child , then it may be merged with its child or parent without affecting the size of any tree cut. The number of vertices not belonging to V is therefore at most $n-1$ (which is the case if the tree is binary and V lies at the leaves), so the number of tree edges is bounded by $2n-2$.

The number of edge sets containing less than $n/8$ edges is at most

$$\sum_{i=1}^{n/8}\binom{2n}{i} < \sum_{i=1}^{n/8}\binom{2n}{n/8} = \frac{n}{8}\binom{2n}{n/8} < \frac{n}{8}(16e)^{n/8} < \frac{n}{8}2^{3n/4} < 2^{3n/4+\log n} \ .$$

The number of vertex sets $S \subset V$, $|S| < n/2$ is larger than 2^{n-2}. As $2^{3n/4+\log n} < 2^{n-2}$ for $n \geq 32$, it follows that there exist sets $S \subset V$ whose corresponding cut in the tree contains at least $n/8$ edges. The cost of these tree cuts is therefore at least $n/8$, while the cut cost in the graph is 1. It follows that the approximation ratio of the tree is $\Omega(n)$. □

3.2.2 Vertex Cut Trees for Hyperedge Cuts. The following theorem gives a lower bound on the quality of vertex cut trees for approximating hyperedge cuts in a hypergraph G over n vertices.

Theorem 7. *There exists a hypergraph on n vertices such that the quality of any vertex-weighted cut tree is $\Omega(\sqrt{n})$.*

Proof. Consider a graph $G = (V, E)$ with a top vertex v connected to n vertices $U = \{u_1, \ldots, u_n\}$ with simple edges of weight 1, as well as a hyperedge of weight $\sqrt{n}$ spanning all vertices u_i. The graph is shown in Figure 2.

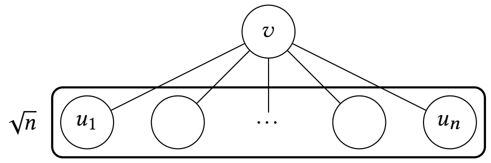

Figure 2: Bad example for approximating hyperedge cuts via vertex cut trees.

The cut size of any set $S \subsetneq U$ is $\delta_G(S) = \sqrt{n} + |S|$, as both the hyperedge and all edges connecting S to the top vertex v must be removed. The cut size of any set $S' \cup \{v\}$, $\emptyset \neq S' \subseteq U$ is $\delta_G(S' \cup \{v\}) = \sqrt{n} + n - |S'|$. It follows that any set S with $1 \leq \ell < n/2$ vertices from U has $\delta_G(S) \geq \sqrt{n} + \ell$.

Let T be a dominating vertex cut tree of G. Fix a root of T so that none of its children have more than $n/2$ vertices from U in its subtree. Also fix an arbitrary order among the children of each node. Every node x of this tree with ℓ vertices from U in its subtree defines a cut in the graph with both sides having at least $\ell/3$ vertices from U (note that this in particular also holds for the root as we can partition the children into two sets such that both sets contain at least $n/3$ vertices from U). As the tree is dominating, node x must have weight at least $\ell/3$.

Suppose without loss of generality that an in-order traversal of the tree visits the vertices of U in the order $u_1, u_2, \ldots, u_n$. Let $k = \lfloor\sqrt{n}\rfloor$ and $S = \{u_1, u_{k+1}, \ldots, u_{k(k-1)+1}\}$. As $|S| = \Theta(\sqrt{n})$, the cut size in G is $\delta_G(S)$ is $\Theta(\sqrt{n})$. We show that the vertex cut cost in the tree $\gamma_T(S, V \setminus S)$ is $\Omega(n)$.

In the following we refer to the vertices in S as S-vertices and to the vertices in $U - S$ as U-vertices. Fix a cut X in T. We first show that the cut cost of X is large if only few U-vertices can reach the root.

Claim 2. *If in a sub-tree T' at least ℓ U-vertices from T' are separated from the root $r_{T'}$ the cut cost* inside *T' is at least $\ell/3$.*

Proof. Proof by induction over the height of T'. The claim clearly holds if T' is just a leaf vertex. If a U-vertex is cut away from the root it means that the leaf vertex is in the cut and the cost is at least $\sqrt{n} \geq 1/3$.

For the induction step consider a root vertex $r_{T'}$ with children $c_1, \ldots, c_s$, and assume that from the subtree rooted at a child c_i we aim to separate ℓ_i U-vertices from the root. If we choose $r_{T'}$ into the cut the cost of the cut inside T' is at least $w(r_{T'}) \geq \sum_i \ell_i/3$. Otherwise, we have to pay at least $\ell_i/3$ in each sub-tree by induction hypothesis, which sums up to the same cost. □

This means that the cost of the tree cut for S is at least $\Omega(n)$ if this cut happens to separate at least 50% of the U-vertices from the root r_T. Now, suppose that the cut separates at most 50% of the U-vertices. Observe, that in this case *all* S-vertices must be separated from the root. For every U-vertex u that is not separated from the root remove the path from u to r_T from T. The removed set does not contain any vertices from X and also no S-vertices.

The removal partitions the tree into pieces. The key observation is that at least $\Omega(\sqrt{n})$ pieces must contain an element from S. To see this observe that if between two (not necessarily consecutive) vertices s and s' from S, there exists a vertex from $U - S$ that is connected to r_T, then s and s' must lie in different pieces. This holds because the LCA of s and s' must lie on the path from r_T to u and is taken out. The vertices from $U - S$ that can reach the root partition the sequence $u_1, \ldots, u_n$ into intervals. In the following we argue that at least $\sqrt{n}/2$ of these intervals contain an element form S. Let k_i, $i \geq 1$ denote the number of intervals that contain i S-vertices. The size of such an interval (number of vertices contained in it) is at least $(i-1)\cdot\sqrt{n}$. Summing the size of all these intervals gives

$$\sum_i k_i(i-1)\sqrt{n} \leq \frac{n+\sqrt{n}}{2} \ , \tag{4}$$

where the inequality follows because the U-vertices that can reach the root are not contained in any interval. On the other hand,

$$\sum_i k_i i \geq \sqrt{n} \tag{5}$$

holds, as the left hand side is the total number of S-vertices. Combining equations 4 and 5 gives that the total number of intervals that contain an S-vertex is $\sum_i k_i \geq (\sqrt{n}-1)/2 = \Omega(\sqrt{n})$.

Since in each piece that contains an element from S we have to select at least one vertex to be in the cut X, we have to choose

$\Omega(\sqrt{n})$ vertices each with cost $\sqrt{n}$. This gives a total cut-cost of $\Omega(n)$. □

Note that the same result holds for unweighted hypergraphs. To see this, replace the weighted hyperedge with $\lfloor\sqrt{n}\rfloor$ different hyperedges, each spanning the n vertices of the set U.

3.2.3 Vertex Cut Trees for Weighted Vertex Cuts. We show that the result from the previous section immediately translates to a lower bound for vertex cut trees. The lower bound is based on the graph $G_H = (V_H, E_H)$ shown in Figure 3. Graph G_H is precisely the graph constructed by applying the reduction from Lemma 7 to the hypergraph G from the previous section. The graph G_H consists of a vertex t of weight $\sqrt{n}$ (representing the hyperedge of G_H), connected to vertices $u_1, \ldots, u_n$ of weight $\sqrt{n}+1$. Each of the u_i is connected to a vertex w_i of weight 1 (representing the edges of G_H). Finally, all w_i are connected to a vertex v of weight n. The graph G_H has $N = 2n+2$ vertices of total weight $2n + n(\sqrt{n}+1) + \sqrt{n} = \Theta(N\sqrt{N})$.

Graph G_H is precisely the graph constructed by applying the reduction from Lemma 7 to the hypergraph from the previous section.

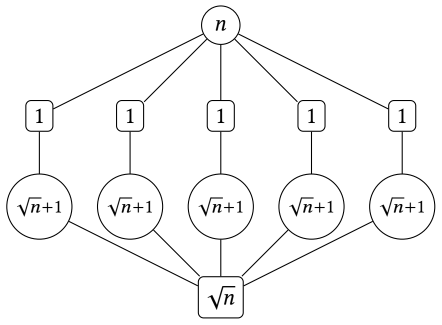

Figure 3: Bad example for approximating weighted vertex cuts via vertex cut trees.

LEMMA 8. *Any dominating, vertex-weighted cut tree T of G_H has quality $\Omega(\sqrt{N})$.*

PROOF. Let $G = (V, E)$ be the hypergraph considered in the previous section, shown in Figure 2. Lemma 7 shows that for any two vertex sets A, B in G, the corresponding cut in G_H has identical cost. As the vertex set of G_H is a superset of the vertex set of G, any vertex cut tree for G_H is a vertex cut tree for G. By Theorem 7, any vertex cut tree for G has quality $\Omega(\sqrt{n})$. As $|V_H| = N = 2n+2$, it follows that any vertex cut tree for G_H has quality $\Omega(\sqrt{n}) = \Omega(\sqrt{N})$.

□

3.2.4 Vertex-Weighted Cut Trees for unweighted Vertex Cuts. We modify the construction shown in Figure 3 as follows. For simplicity of exposition, we change the vertex weight of $\sqrt{n}+1$ to $\sqrt{n}$ and assume that n is square. Replace any vertex of weight w with a clique of w vertices and replace the edges between formerly weighted vertices with a full bipartite graph between the respective cliques. Let U_i denote the clique resulting from the vertex u_i in the weighted graph.

We plan to adapt the proof of Theorem 7. This proof is based on the observation that for any tree, there must exist $\sqrt{n}$ vertices u_i that are *spread apart* in the tree and, therefore, separating these vertices in the tree is costly. Applying the same idea to the unweighted case is not straightforward, as the vertices u_i are now cliques U_i, whose vertices may not be concentrated in some vertex cut tree.

In the unweighted graph, call the clique vertices U_i *core vertices*; enumerate them so that core vertices of the same clique are consecutive. Given any vertex cut tree, an in-order traversal of the tree induces a *permutation* of the core vertices. We show first show that, for any permutation, there is a choice of core vertices from $\Theta(\sqrt{n})$ cliques, such that the following holds: After applying the permutation, there exist $\Theta(\sqrt{n})$ disjoint intervals of length n, in which $\Theta(\sqrt{n})$ chosen core vertices are found. This means that we can choose cliques such that the vertices from these cliques are sufficiently spread apart in the tree.

We formalize the technical observation as follows. Divide the set $L = \{1, \ldots, n\sqrt{n}\}$ into n *clusters* $\{(i-1)\sqrt{n}+1, \ldots, \sqrt{n}\}$ for $i = 1, \ldots, \sqrt{n}$, i.e., a cluster are $\sqrt{n}$ consecutive numbers. Given some permutation π of L, a *group* is a set $\{\pi((j-1)n+1), \ldots, \pi(jn)\}$ for $j = 1, \ldots, \sqrt{n}$, i.e., the groups are n numbers that are consecutive after applying the permutation.

LEMMA 9. *For any permutation π, there is a choice of $2\sqrt{n}$ clusters, such that there are at least $\sqrt{n}/9$ groups from which $\Theta(\sqrt{n})$ elements have been chosen.*

PROOF. Choose $2\sqrt{n}$ clusters uniformly at random and let Y_j be the number of chosen elements in group j. With X_j being the indicator variable of the event that $\sqrt{n}/2 \le Y_j \le 7\sqrt{n}/2$, we show that the expected number of groups with $\Theta(\sqrt{n})$ elements, $\mathbb{E}[\sum_{j=1}^{\sqrt{n}} X_j]$ is at least $\sqrt{n}/9$.

The expectation of Y_j is $2\sqrt{n}$. To see this, let c_{ij} be the total number of elements from cluster i whose group is j, after applying the permutation π. Let C_i be the indicator variable of the event that cluster i is chosen; note that $\mathbb{E}[C_i]$ equals $2\sqrt{n}/n = 2/\sqrt{n}$. Furthermore, the total number of elements in group j is n, thus, $\sum_{i=1}^{n} c_{ij} = n$. With $Y_j = \sum_{i=1}^{n} c_{ij} C_i$ it follows that $\mathbb{E}[Y_j] = 2n/\sqrt{n} = 2\sqrt{n}$.

We now argue that the variance of Y_j is at most $2n$. With Cov denoting the covariance,

$$\mathrm{Var}[Y_j] = \sum_{i=1}^{n} c_{ij}^2 \,\mathrm{Var}[C_i] + \sum_{1\le k<\ell\le n} c_{kj} c_{\ell j} \,\mathrm{Cov}[C_k, C_\ell] \ .$$

The variance of indicator variable C_i is $\frac{2}{\sqrt{n}}(1 - \frac{2}{\sqrt{n}}) = (2\sqrt{n}-4)/n$. The covariance of C_k and C_ℓ is $\mathbb{E}[C_k \cdot C_\ell] - \mathbb{E}[C_k]\mathbb{E}[C_\ell]$. The expectation of $C_k \cdot C_\ell$ is the probability that clusters k and ℓ are both chosen. As their common choice is less likely as the probability of the two being chosen in an independent choice, $\mathbb{E}[C_k \cdot C_\ell] \le \mathbb{E}[C_k] \cdot \mathbb{E}[C_\ell]$. It follows that the covariance is negative and

$$\mathrm{Var}[Y_j] \le \sum_{i=1}^{n} c_{ij}^2 \,\mathrm{Var}[C_i] < \frac{2}{\sqrt{n}} \sum_{i=1}^{n} c_{ij}^2 \ .$$

The sum $\sum_{i=1}^{n} c_{ij}^2$ is maximized if $\sqrt{n}$ of the terms are at the maximum of $\sqrt{n}$, thus $\sum_{i=1}^{n} c_{ij}^2 \leq n\sqrt{n}$. Plugging this fact in the above equation gives that $\mathrm{Var}[Y_j] \leq n$.

Chebychev's inequality states with $\mu = \mathbb{E}[Y_j]$ and $\sigma^2 = \mathrm{Var}[Y_j]$ for any $k > 0$

$$\Pr\left[|Y_j - \mu| \leq k\right] \geq 1 - \frac{\sigma^2}{k^2} \ .$$

We have $\sigma^2 \leq 2n$, $\mu = 2\sqrt{n}$ and we choose $k = 3\sqrt{n}/2$. It follows that this implies that

$$\Pr\left[\sqrt{n}/2 \leq Y_j \leq 7\sqrt{n}/2\right] \geq 1 - \frac{2n}{(3\sqrt{n}/2)^2} = 1 - \frac{8}{9} = \frac{1}{9} \ .$$

The expectation of X_j is therefore at least $1/9$ and consequently $\mathbb{E}[\sum_{j=1}^{\sqrt{n}} X_j]$ is at least $\sqrt{n}/9$. □

We sketch the proof of the theorem. Note that the strategy is very similar to the proof of Theorem 7, with the modifications highlighted at the beginning of this section.

Theorem 8. *Any dominating, vertex-weighted cut tree T of G has quality $\Omega(N^{1/3})$, where N is the number of vertices of G.*

Proof Sketch. The cut size of any set of ℓ cliques U_i is $\sqrt{n}+\ell$, as the clique representing the hyperedge and the vertices representing the simple edges must be removed.

Let T be a dominating vertex cut tree of G. Fix a root of T so that none of its children have more than $n\sqrt{n}/2$ core vertices in its subtree. We show that every node x of this tree with $\ell > \sqrt{n}$ core vertices in its subtree must have weight at least $\max\{\sqrt{n}, \ell/3\sqrt{n}\}$. Node x defines a cut in the graph, with both sides having at least $\ell/3$ core vertices. If this cut divides a clique, the cut cost per vertex for this clique is at least one, which implies the theorem. If no cliques are divided, then the cut cost is $\sqrt{n} + \ell/3 > \ell/3\sqrt{n}$, as shown above.

Enumerate the core vertices of the tree via in-order traversal. The vertices $jn+1, \ldots, (j+1)n$ form a *group* in the tree, for $j = 1, \ldots, \sqrt{n}$. Choose $2\sqrt{n}$ cliques U_i so that $\Theta(\sqrt{n})$ groups have $\Theta(\sqrt{n})$ chosen. These vertices form the set S. An appropriate choice is possible, as shown by Lemma 9. The cut cost of S in the graph is $\Theta(\sqrt{n})$, as argued above. We prove that the cut cost of S in the tree is $\Omega(n)$.

In the following, call S-vertices the vertices in S and U-vertices those in $U - S$.

We claim that if in a subtree T' at least ℓ U-vertices are separated from the root, the cut cost inside T' is at least $\ell/(3\sqrt{n})$. This can be shown by induction as in Claim 2.

Fix a cut X in the tree. If the cut X separates at least $n\sqrt{n} - 3n$ U-vertices from the root (so all but n U-vertices), then the cut cost is $\Omega(n)$ by the above claim. Now, suppose that the cut separates at most n of the U-vertices from the root. For every U-vertex that is not separated from the root remove the path from u to the root from T. The removed set does not contain any vertices from X and also no S-vertices.

The removal partitions the tree into pieces. As $\Theta(\sqrt{n})$ groups contain $\Theta(\sqrt{n})$ S-vertices, there exist pieces in all of these groups. However, if an interval has size $z \leq \sqrt{n}$, the cost of the corresponding piece will be at least z (as is the cut cost in the graph). On the other hand, an interval can contain at most $2\sqrt{n}$ S-vertices, as the S-vertices are n core vertices apart and only n U-vertices to fill the intervals exist. It follows that the cost per S-vertex is at least $1/2$, so the total cost must be $\Omega(n)$.

As the total number of vertices N is $\Theta(n^{3/2})$, the gap between $\gamma_G(S)$ and $\gamma_T(S)$ is $\Omega(\sqrt{n}) = \Omega(N^{1/3})$. □

4 ACKNOWLEDGEMENTS

The authors would like to thank the anonymous referees for helpful comments. The work of Roy Schwartz is supported by GIF grant I-2439-407.6/2016 and ISF grant 1336/16.

REFERENCES

[1] Konstantin Andreev and Harald Räcke. 2004. Balanced Graph Partitions. In *Proc. of the 16th SPAA.* 120–124. https://doi.org/10.1145/1007912.1007931
[2] Sanjeev Arora, Satish Rao, and Umesh Vazirani. 2004. Expander Flows, Geometric Embeddings, and Graph Partitionings. In *Proc. of the 36th STOC.* 222–231. https://doi.org/10.1145/1007352.1007355
[3] Nikhil Bansal, Uriel Feige, Robert Krauthgamer, Konstantin Makarychev, Viswanath Nagarajan, Joseph (Seffi) Naor, and Roy Schwartz. 2011. Min-Max Graph Partitioning and Small Set Expansion. In *Proc. of the 52nd FOCS.* 17–26. https://doi.org/10.1109/FOCS.2011.79
[4] Eden Chlamtáč, Michael Dinitz, and Robert Krauthgamer. 2012. Everywhere-Sparse Spanners via Dense Subgraphs. In *Proc. of the 53th FOCS.* 758–767. https://doi.org/10.1109/FOCS.2012.61
[5] Eden Chlamtáč, Michael Dinitz, and Yury Makarychev. 2017. Minimizing the Union: Tight Approximations for Small Set Bipartite Vertex Expansion. In *Proc. of the 28th SODA.* SIAM, 881–899. https://doi.org/10.1137/1.9781611974782.56
[6] Uriel Feige, MohammadTaghi Hajiaghayi, and James R. Lee. 2008. Improved Approximation Algorithms for Minimum Weight Vertex Separators. *SIAM J. Comput.* 38, 2 (2008), 629–657. https://doi.org/10.1137/05064299X
[7] Uriel Feige, Robert Krauthgamer, and Kobbi Nissim. 2000. Approximating the Minimum Bisection Size. In *Proc. of the 32nd STOC.* 530–536. https://doi.org/10.1145/335305.335370
[8] Andreas E. Feldmann and Luca Foschini. 2012. Balanced Partitions of Trees and Applications. In *Proc. of the 29th STACS.* 100–111. https://doi.org/10.4230/LIPIcs.STACS.2012.100
[9] Chris Harrelson, Kirsten Hildrum, and Satish Rao. 2003. A Polynomial-time Tree Decomposition to Minimize Congestion. In *Proceedings of the Fifteenth Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA '03).* 34–43.
[10] Bruce Hendrickson and Tamara G. Kolda. 2000. Graph Partitioning Models for Parallel Computing. *Parallel Comput.* 26, 12 (Nov. 2000), 1519–1534. https://doi.org/10.1016/S0167-8191(00)00048-X
[11] Robert Krauthgamer, Joseph (Seffi) Naor, and Roy Schwartz. 2009. Partitioning Graphs into Balanced Components. In *Proc. of the 20th SODA.* SIAM, 942–949. arXiv:soda:1496770.1496872
[12] Frank Thomson Leighton and Satish B. Rao. 1999. Multicommodity Max-Flow Min-Cut Theorems and Their Use in Designing Approximation Algorithms. *J. ACM* 46, 6 (1999), 787–832. https://doi.org/10.1145/331524.331526
[13] Anand Louis and Yury Makarychev. 2014. Approximation Algorithms for Hypergraph Small Set Expansion and Small Set Vertex Expansion. In *Proc. of the 17th APPROX/RANDOM.* 339–355. https://doi.org/10.4230/LIPIcs.APPROX-RANDOM.2014.339
[14] Pasin Manurangsi. 2017. Almost-Polynomial Ratio ETH-Hardness of Approximating Densest k-Subgraph. In *Proc. of the 49th STOC.* 954–961. https://doi.org/10.1145/3055399.3055412
[15] David A. Papa and Igor L. Markov. 2007. Hypergraph Partitioning and Clustering. In *In Approximation Algorithms and Metaheuristics.*
[16] Harald Räcke. 2002. Minimizing Congestion in General Networks. In *43rd Symposium on Foundations of Computer Science (FOCS).* 43–52.
[17] Harald Räcke. 2008. Optimal Hierarchical Decompositions for Congestion Minimization in Networks. In *Proc. of the 40th STOC.* 255–264.
[18] Harald Räcke and Chintan Shah. 2014. Improved Guarantees for Tree Cut Sparsifiers. In *Proc. of the 22nd ESA.* 774–785. https://doi.org/10.1007/978-3-662-44777-2_64
[19] Harald Räcke and Richard Stotz. 2016. Improved Approximation Algorithms for Balanced Partitioning Problems. In *Proc. of the 33rd STACS.* 58:1–58:14. https://doi.org/10.4230/LIPIcs.STACS.2016.58
[20] S. Rajamanickam and E. G. Boman. 2013. Parallel Graph Partitioning with Zoltan: Is hypergraph partitioning worth it? In *Graph Partitioning and Graph Clustering,* D. A. Bader, H. Meyerhenke, P. Sanders, and D. Wagner (Eds.). AMS Contemporary Mathematics, Vol. 588. AMS, 37–52.