Approximating Sparsest Cut in Low-Treewidth Graphs via Combinatorial Diameter

PARINYA CHALERMSOOK, Aalto University, Finland
MATTHIAS KAUL and MATTHIAS MNICH, Hamburg University of Technology, Institute for Algorithms and Complexity, Germany
JOACHIM SPOERHASE, University of Sheffield, United Kingdom
SUMEDHA UNIYAL, Aalto University, Finland
DANIEL VAZ, École Normale Supérieure Paris and Université Paris Cité, CNRS, IRIF, France

The fundamental Sparsest Cut problem takes as input a graph G together with edge capacities and demands, and seeks a cut that minimizes the ratio between the capacities and demands across the cuts. For n-vertex graphs G of treewidth k, Chlamtáč, Krauthgamer, and Raghavendra (APPROX 2010) presented an algorithm that yields a factor-2^{2^k} approximation in time $2^{O(k)} \cdot n^{O(1)}$. Later, Gupta, Talwar and Witmer (STOC 2013) showed how to obtain a 2-approximation algorithm with a blown-up run time of $n^{O(k)}$. An intriguing open question is whether one can simultaneously achieve the best out of the aforementioned results, that is, a factor-2 approximation in time $2^{O(k)} \cdot n^{O(1)}$.

In this paper, we make significant progress towards this goal, via the following results:

(i) A factor-$O(k^2)$ approximation that runs in time $2^{O(k)} \cdot n^{O(1)}$, directly improving the work of Chlamtáč et al. while keeping the run time single-exponential in k.
(ii) For any $\varepsilon \in (0, 1]$, a factor-$O(1/\varepsilon^2)$ approximation whose run time is $2^{O(k^{1+\varepsilon}/\varepsilon)} \cdot n^{O(1)}$, implying a constant-factor approximation whose run time is nearly single-exponential in k and a factor-$O(\log^2 k)$ approximation in time $k^{O(k)} \cdot n^{O(1)}$.

Key to these results is a new measure of a tree decomposition that we call *combinatorial diameter*, which may be of independent interest.

ACM Reference Format:
Parinya Chalermsook, Matthias Kaul, Matthias Mnich, Joachim Spoerhase, Sumedha Uniyal, and Daniel Vaz. 2024. Approximating Sparsest Cut in Low-Treewidth Graphs via Combinatorial Diameter. *ACM Trans. Algor.* 0, 0, Article 0 (January 2024), 20 pages. https://doi.org/10.1145/3632623

1 INTRODUCTION

In the Sparsest Cut problem, we are given a graph together with capacities and demands on the edges, and our goal is to find a cut that minimizes the ratio between the capacities and demands across the cut. Formally, we define it as follows:

Problem Definition. In the Sparsest-Cut problem (with general demands), the input is a graph $G = (V, E_G)$ with positive edge capacities $\{\text{cap}_e\}_{e \in E_G}$ and a demand graph $D = (V, E_D)$ (on the same set of vertices) with positive demand values $\{\text{dem}_e\}_{e \in E_D}$. The aim is to compute the values

$$\Phi_{G,D} := \min_{S \subseteq V} \Phi_{G,D}(S), \qquad \Phi_{G,D}(S) := \frac{\sum_{e \in E_G(S, V-S)} \text{cap}_e}{\sum_{e \in E_D(S, V-S)} \text{dem}_e}.$$

Authors' addresses: Parinya Chalermsook, parinya.chalermsook@aalto.fi, Aalto University, Espoo, Finland; Matthias Kaul, matthias.kaul@tuhh.de; Matthias Mnich, matthias.mnich@tuhh.de, Hamburg University of Technology, Institute for Algorithms and Complexity, Hamburg, Germany; Joachim Spoerhase, j.spoerhase@sheffield.ac.uk, University of Sheffield, United Kingdom; Sumedha Uniyal, Aalto University, Espoo, Finland; Daniel Vaz, École Normale Supérieure Paris and Université Paris Cité, CNRS, IRIF, Paris, France, daniel.vaz@ens.fr.

2024. 1549-6325/2024/1-ART0 $15.00
https://doi.org/10.1145/3632623

To avoid division by zero, we only consider cuts S for which $\sum_{e \in E_D(S,V-S)} \text{dem}_e$ is non-zero. The value $\Phi_{G,D}(S)$ is called the *sparsity* of the cut S.

Sparsest-Cut is among the most fundamental optimization problems that has attracted the attention of both computer scientists and mathematicians. As the problem is NP-hard [26], the focus has been to study approximation algorithms for the problem. Over the past four decades, several breakthrough results have eventually culminated in a factor-$\tilde{O}(\sqrt{\log n})$ approximation in polynomial time [1, 2, 24]. On the lower-bound side, the problem is APX-hard [12] and, assuming the Unique Games Conjecture, does not admit any constant-factor approximation in polynomial time [8].

The significance of the Sparsest-Cut problem stems from both applications and mathematical reasons. From the point of view of applications, the question of partitioning the universe into two parts while minimizing the "loss" across the interface is crucial in any divide-and-conquer approach e.g., in image segmentation. From a mathematical/geometric viewpoint, the integrality gap of convex relaxations for sparsest cuts is equivalent to the embeddability of any finite metric space (for LP relaxation) and of any negative-type metric (for SDP relaxation)[1] into ℓ_1. Therefore, it is not a surprise that this problem has attracted interest from both computer science and mathematics (geometry, combinatorics, and functional analysis) communities.

In 2010, Chlamtáč, Krauthgamer, and Raghavendra [10] initiated the study of sparsest cuts in the low-treewidth regime, which restricts the treewidth of the capacitated subgraph G to some integer k (no restrictions are placed on the demand graph D). Chlamtáč et al. devised a factor-2^{2^k} approximation algorithm (CKR) that runs in time $2^{O(k)} \cdot n^{O(1)}$, with k being the treewidth of the capacitated subgraph G . Later, Gupta, Talwar and Witmer [18] showed how to obtain a factor-2 approximation (GTW) with a blown-up run time of $n^{O(k)}$; they further showed that there is no $(2-\varepsilon)$-approximation for any $\varepsilon > 0$ on constant-treewidth graphs, assuming the Unique Games Conjecture. Cohen-Addad, Mömke, and Verdugo [15] recently gave a factor-2 approximation algorithm (CMV) with run time $2^{2^{O(k)}} \cdot n^{O(1)}$, which removes the dependency on k in the exponent of n, but suffers from a doubly-exponential dependence on k.

It remains an intriguing open question whether one can simultaneously achieve the best run time and approximation factor. In particular, in this paper we address the following question:

Does Sparsest-Cut admit a factor-2 approximation algorithm with run time $2^{O(k)} \cdot n^{O(1)}$?

Broader perspectives. Given the relevance of sparsest cuts, significant effort has been invested into understanding when Sparsest-Cut instances are "easy". In trees, optimal sparsest cuts can be found in polynomial time (see e.g., Gupta et al. [22]). For many other well-known graph classes, finding optimal sparsest cuts is NP-hard, and thus researchers attempted to find constant-factor approximation algorithms in polynomial time. They have succeeded, over the past two decades, for several classes of graphs, such as outerplanar, ℓ-outerplanar, bounded-pathwidth and bounded-treewidth graphs [9, 10, 17, 18, 23], as well as planar graphs [13].

As mentioned earlier, sparsest cuts are not only interesting from the perspective of algorithm design, but also from the perspectives of geometry, probability theory and convex optimization. Indeed, the famous conjecture of Gupta, Newman, Rabinovich, and Sinclair [17] postulates that any minor-free graph metric embeds into ℓ_1 with constant distortion, which would imply that all such graphs admit a constant approximation for the Sparsest-Cut problem. The conjecture has been verified in various graph classes [9, 23], but remains open even for bounded-treewidth graph families.

[1]A metric (X, d) is said to be *negative type*, if $(X, \sqrt{d})$ embeds isometrically into a Hilbert space.

To us, perhaps the most interesting aspect of the treewidth parameter [10, 18] is its connection to the power of hierarchies of increasingly tight convex relaxations of combinatorial optimization problems (see, for instance, the work by Laurent [21]). In the bounded-treewidth setting, a (problem-independent!) LP rounding algorithm performs surprisingly well for many combinatorial optimization problems: not only does it achieve optimal solutions for various fundamental problems [3, 25, 28], but it has also led to tight approximation factors for problems such as Group Steiner Tree [6, 7, 16, 19]. In this way, for these aforementioned problems, such a problem-oblivious LP rounding algorithm provides a natural framework to generalize an optimal algorithm on trees to nearly-optimal ones on low (perhaps super-constant) treewidth graphs. Our work can be seen as trying to develop such understanding in the context of Sparsest-Cut.

1.1 Our Results

We present several results that may be seen as an intermediate step towards the optimal result. Our main technical results are summarized in the following theorem.

Theorem 1.1. *For the following functions t and α, there are algorithms that run in time $t(k) \cdot n^{O(1)}$ and achieve approximation factors $\alpha(k)$ for the* Sparsest-Cut *problem on graphs of treewidth k:*

- $t(k) = 2^{O(k)}$ *and* $\alpha(k) = O(k^2)$.
- $t(k) = 2^{O(k^2)}$ *and* $\alpha(k) = O(1)$.
- *For any* $\varepsilon \in (0, 1]$, $t(k) = \exp\left(O\left(\frac{k^{1+\varepsilon}}{\varepsilon}\right)\right)$ *and* $\alpha(k) = O(1/\varepsilon^2)$.

For the proof, we refer to Sections 4.1 to 4.3 for each of the respective constructions.

Our first result directly improves the approximation factor of 2^{2^k} by Chlamtáč et al., while keeping the run time single-exponential in k. Our second result shows that, with only slightly larger run time, one can achieve a constant approximation factor. Compared to Gupta et al., our result has a constant blowup in the approximation factor (independent of k), but has a much better run time ($2^{O(k^2)}$ instead of $n^{O(k)}$); compared to Chlamtáč et al., our result has a much better approximation factor ($O(1)$ instead of 2^{2^k}), while maintaining nearly the same asymptotic run time.

Finally, our third result gives us a range of different results trading off the size of the approximation factor against the run time of the algorithm. We remark that, by plugging in $\varepsilon = \Omega(1/\log k)$, we obtain a factor-$O(\log^2 k)$ approximation in time $k^{O(k)} \cdot n^{O(1)}$.

1.2 Overview of Techniques

In this section, we sketch the main ideas used in deriving our results. We assume certain familiarity with the notions of treewidth and tree decomposition, however for completeness the formal definition of tree decompositions is restated in Section 2. Let $(\mathcal{T}, \mathcal{B})$ be a tree decomposition of a graph G, with $\mathcal{B}$ being a collection of bags $B_t \subseteq V(G)$ for all $t \in V(\mathcal{T})$. The width of $(\mathcal{T}, \mathcal{B})$ is $w(\mathcal{T}, \mathcal{B}) = \max_{i \in V(\mathcal{T})} |B_i| - 1$, and the *treewidth* of G is the minimum width over all tree decompositions of G.

The run times of algorithms that deal with the treewidth parameter generally depend on $w(\mathcal{T}, \mathcal{B})$, so when designing an algorithm in low-treewidth graphs G one usually starts with a near-optimal tree decomposition $(\mathcal{T}, \mathcal{B})$, satisfying $w(\mathcal{T}, \mathcal{B}) = O(k)$, where k is the treewidth of G. As an example, the CKR algorithm [11][2] for Sparsest-Cut runs in time $2^{O(w(\mathcal{T},\mathcal{B}))} \cdot n^{O(1)}$ and gives approximation factor $2^{2^{w(\mathcal{T})}}$. In this setting, it is always desirable to use a tree decomposition of lowest possible width, even though an increase in the width would still allow us to obtain the

[2]In the remainder of the paper, we cite the arXiv version of the paper by Chlamtáč et al. [10], as we need some results that are only present in that version.

desired run time of $2^{O(k)} \cdot n^{O(1)}$. In particular, with width $w(\mathcal{T}, \mathcal{B}) = O(\log n) + \beta(k)$ the CKR algorithm runs in time $2^{O(\beta(k))} \cdot n^{O(1)}$, but it achieves a poor approximation ratio.

We aim to exploit this flexibility in the run time component by devising a refined analysis of the CKR algorithm so that its approximation factor depends on the diameter d of $\mathcal{T}$. We can then increase the width of a given decomposition while decreasing its diameter to obtain trade-offs between the runtime and approximation factor. In hindsight, this analysis is already present in the original work of Chlamtáč et al. and gives an approximation factor of $O(d^2)$, although it is not useful when used naïvely, since d will be $\Omega(\log n)$. However, we can use it by modifying the tree decomposition to obtain a trade-off between runtime and approximation guarantee.

For a simplified example of how to obtain such a trade-off, consider a path decomposition $(\mathcal{P}, \mathcal{B})$. If we combine pairs of bags along the path we will obtain a new path decomposition whose width is twice that of $(\mathcal{P}, \mathcal{B})$, but its diameter has been halved. Taking this further, if $\mathcal{P}$ had diameter $O(\log n)$ we could combine subpaths of $\log n / w(\mathcal{P}, \mathcal{B})$ many bags to obtain a modified decomposition of width $\log n + w(\mathcal{P}, \mathcal{B})$ and diameter $O(w(\mathcal{P}, \mathcal{B}))$.

Let us now go from this simple example to the general setting, where instead of path decompositions we deal with tree decompositions. Initially, we obtain a tree decomposition $(\mathcal{T}, \mathcal{B})$ of G of width $O(k)$ and diameter $O(\log n)$ by running an algorithm by Korhonen [20] and subsequently applying an algorithm of Bodlander [4], in time $2^{O(k)} \cdot O(n)$. Combining such a tree decomposition with the aforementioned bag composition argument would already be sufficient to greatly improve the approximation factor of Chlamtáč et al., if it could be extended to tree rather than path decompositions. While it is possible to reduce the analysis of the CKR algorithm to the case of path decompositions, one cannot simply combine sets of neighbouring bags in the case of tree decompositions: since nodes may have many neighbours, this would cause too large an increase in the width.

To overcome this impediment, we introduce the concept of "combinatorial diameter" of a tree decomposition. Informally, the combinatorial length of a path between u and v in $\mathcal{T}$ measures the number of "non-redundant bags" that lie on the unique path in $\mathcal{T}$ connecting the bags of u and v. We say that the combinatorial diameter $\Delta(\mathcal{T}, \mathcal{B})$ of $(\mathcal{T}, \mathcal{B})$ is the maximum combinatorial length of any path in $\mathcal{T}$. We refer to Section 3.1 for formal definitions. The notion of combinatorial diameter allows us to argue that, by purposefully modifying a tree decomposition, many of the nodes do not impact the approximation factor (i.e. are redundant) since their bags contain vertices that occur only briefly during rounding, and thus are in some sense unable to carry information forward that could degrade performance. Formally, Theorem 3.6 shows that the approximation factor of the CKR algorithm can be upper bounded, in terms of the combinatorial diameter, by $\min\{O(\Delta(\mathcal{T}, \mathcal{B})^2), 2^{2^{w(\mathcal{T}, \mathcal{B})}}\}$. Moreover, in the special case of $\Delta(\mathcal{T}, \mathcal{B}) = 1$, the CKR algorithm gives a 2-approximation, which follows from the arguments of Gupta et al. [18].

With this result at hand it then suffices to construct a tree decomposition $(\mathcal{T}, \mathcal{B})$ with simultaneously low $w(\mathcal{T}, \mathcal{B})$ and low $\Delta(\mathcal{T}, \mathcal{B})$ to obtain a good approximation algorithm, i.e. one that achieves a $O(\Delta(\mathcal{T}, \mathcal{B})^2)$-approximation in time $2^{w(\mathcal{T}, \mathcal{B})} \cdot n^{O(1)}$. Constructing such tree decompositions is possible using a generalised version of the width/diameter trade-off sketched for path decompositions, now exploiting the fact that we only need to decrease the combinatorial diameter, while the actual diameter may stay large.

This framework is surprisingly powerful, and allows us to construct a range of different tree decompositions exhibiting different widths and combinatorial diameters, which we summarise in Table 1. Furthermore, we can also interpret the existing results by CKR [11], GTW [18], and CMV [14] in our framework as giving constructions of tree decompositions with different trade-offs between width and combinatorial diameter. Under this lens, the existing work on approximation

	Lemma 4.3	Lemma 4.5	Lemma 4.7 $\forall q \in \mathbb{N}_1$	GTW [18]	CMV [14]
$\Delta(\mathcal{T}, \mathcal{B})$	$O(k)$	3	$2q+1$	1	1
$w(\mathcal{T}, \mathcal{B})$	$O(\log n + k)$	$O(\log n + k^2)$	$O(\log n + qk^{1+1/q})$	$O(k \log n)$	$O(\log n) + 2^{O(k)}$

Table 1. A summary of achievable trade-offs between width and combinatorial diameter of a tree decomposition for a graph with n vertices and treewidth k. All of these constructions give rise to $O(\Delta(\mathcal{T}, \mathcal{B})^2)$-approximation algorithms running in time $2^{w(\mathcal{T},\mathcal{B})} \cdot n^{O(1)}$. For $\Delta(\mathcal{T}, \mathcal{B}) = 1$ the approximation factor can be shown to be 2, by the arguments in GTW [18].

algorithms for the Sparsest-Cut problem in the bounded-treewidth regime can then be understood as being applications of the CKR algorithm to tree decompositions with the correct compromise between width and combinatorial diameter.

2 PRELIMINARIES

We use $\mathbb{N}_1$ to denote the positive integers $\{1, 2, \dots\}$ and $\mathbb{N}_0$ to mean $\mathbb{N}_1 \cup \{0\}$.

Tree decompositions. Let G be a graph. A tree decomposition $(\mathcal{T}, \mathcal{B})$ of G is a tree $\mathcal{T}$ together with a collection $\mathcal{B} = \{B_i\}_{i \in V(\mathcal{T})}$ of *bags*, where the bags $B_i \subseteq V(G)$ satisfy the following properties:

- $V(G) = \bigcup_{i \in V(\mathcal{T})} B_i$.
- For each edge $uv \in E(G)$, there is a bag B_i containing both u and v.
- For each vertex $v \in V(G)$, the collection of bags containing v induces a subtree of $\mathcal{T}$.

The *width* of $(\mathcal{T}, \mathcal{B})$ is given by $w(\mathcal{T}, \mathcal{B}) = \max_{i \in V(\mathcal{T})} |B_i| - 1$, and the *treewidth* of G is the minimum width of any tree decomposition of G.

Korhonen [20] shows how to compute a tree decomposition of a treewidth-k graph that has width $2k$ in time $2^{O(k)} \cdot n$, and a procedure due to Bodlaender [4] allows us to transform any such decomposition into one that has diameter $O(\log n)$ and width at most $6k + 2$.

We will also assume that the number of nodes in the tree $\mathcal{T}$ is bounded by $O(n)$, by the following simple process: repeatedly delete any bag that has at most one child and is a subset of its parent; after this process ends, there are at most n bags with at most one child (since each must introduce a new vertex), and thus at most $2n$ nodes in total.

We generally use r to denote the root of $\mathcal{T}$, and $p : V(\mathcal{T}) \to V(\mathcal{T})$ for the parent of a node with respect to root r, where $p(r) = r$. We sometimes refer to $B_{p(i)}$ as the *parent bag* of B_i. Let $\mathcal{T}_{i \leftrightarrow j}$ be the set of nodes on the unique path in tree $\mathcal{T}$ between nodes $i, j \in V(\mathcal{T})$ (possibly $i = j$). For a set $X \subseteq V(\mathcal{T})$ of nodes, we use the shorthand $B(X) = \bigcup_{i \in X} B_i$ (the union of bags for nodes in X).

We will treat cuts in a graph as assignments of $\{0, 1\}$ to each vertex, which we formally denote as X-assignments:

Definition 2.1. Let X be some finite set. An *X-assignment* is a map $f : X \to \{0, 1\}$. We denote by $\mathcal{F}[X]$ the set of all X-assignments. For some probability distribution μ over $\mathcal{F}[X]$ and set $Y \subseteq X$ we define $\mu|_Y$ to be the distribution given by

$$\Pr_{f \sim \mu|_Y}[f = f'] = \Pr_{f \sim \mu}[f|_Y = f'] \quad \forall f' \in \mathcal{F}[Y] .$$

3 ALGORITHM AND COMBINATORIAL DIAMETER

Our approach is based on the new relation between the algorithm of Chlamtáč et al. [11] and our novel notion of "combinatorial diameter". In Section 3.1, we present the definition of combinatorial

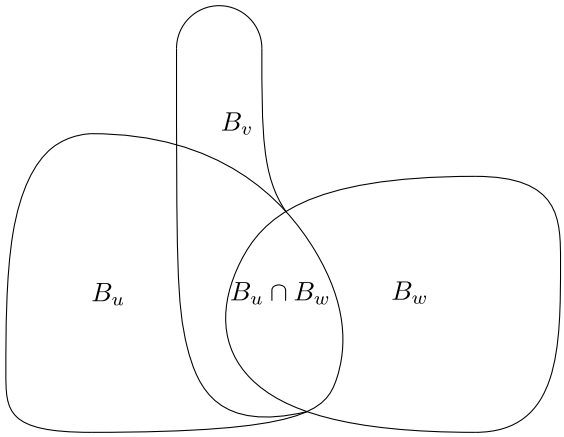

Fig. 1. Illustration of the bags of a redundant node v and its neighbors. Notice that the vertices in B_v are either in B_u, in which case they are already processed at u, or they are neither in B_u nor in B_w, in which case they have no impact on the rounding algorithm along this path.

diameter. The subsequent sections give the description of Chlamtáč et al. and prove its connection to the combinatorial diameter.

3.1 Our New Concept: Combinatorial Diameter

Let G be a graph and let $(\mathcal{T}, \mathcal{B})$ be a tree decomposition of G, where $\mathcal{B} = \{B_i\}_{i \in V(\mathcal{T})}$.

Definition 3.1 (Redundant bags). Fix $s, t \in V(\mathcal{T})$. Let $v \in V(\mathcal{T}) \setminus \{s, t\}$ be a node with neighbors u and w on the path $\mathcal{T}_{s \leftrightarrow t}$. When $B_v \cap B_w \subseteq B_u$, we say that v is (s, t)-*redundant*.

Intuitively, each redundant node v can be thought of as a subset of u, since the vertices in $B_v \setminus B_u$ occur only in B_v within $\mathcal{T}_{s \leftrightarrow t}$. As a consequence, we can show that they do not affect the rounding behaviour of the CKR algorithm with respect to s and t (therefore "redundant"). It might be surprising that redundancy depends on the direction of the traversal, i.e. a node might be (s, t)-redundant but not (t, s)-redundant. We will be able to circumvent this by observing that the CKR algorithm is not impacted by the choice of direction, so the definition will effectively be symmetric in the sense that if a node is redundant in one of the directions, we may assume that the algorithm is using this direction.

Definition 3.2 (Simplification). Let $(\mathcal{T}, \mathcal{B})$ be a tree decomposition, and let $s, t \in V(\mathcal{T})$. A *simplification* of $\mathcal{T}_{s \leftrightarrow t}$ is a path P which can be generated from $\mathcal{T}_{s \leftrightarrow t}$ by repeatedly applying the following rule:

> Delete an (s, t)-redundant node v with neighbors u, w on the path $\mathcal{T}_{s \leftrightarrow t}$, and add the edge $\{u, w\}$. We call this operation *bypassing* v.

Definition 3.3 (Combinatorial diameter). Let $(\mathcal{T}, \mathcal{B})$ be a tree decomposition, and let $s, t \in V(\mathcal{T})$. We say $\mathcal{T}_{s \leftrightarrow t}$ has *combinatorial length* at most ℓ if it has a simplification of length at most ℓ. The *combinatorial diameter* of $(\mathcal{T}, \mathcal{B})$ is defined as the minimum δ such that, for all u, v, the path $\mathcal{T}_{u \leftrightarrow v}$ has combinatorial length at most δ.

Algorithm 1: Algorithm SC-Round

Data: $G, (\mathcal{T}, \{B_i\}_{i \in V(\mathcal{T})}), \{\mu_L\}$

Start at any bag B_0, sample $f|_{B_0}$ from μ_{B_0};

We process the bags in non-decreasing order of distance from B_0 ;

foreach *bag B with a processed parent bag B'* **do**

Let $B^+ = B \cap B'$ the subset of B on which f is fixed. Let $B^- := B \setminus B^+$. Sample $f|_{B^-}$ according to

$$\Pr[f|_{B^-} = f'] = \Pr_{f^* \sim \mu_B}[f^*|_{B^-} = f' \mid f^*|_{B^+} = f|_{B^+}] \quad \forall f' \in \mathcal{F}[B^-]$$

end

Result: f

3.2 Algorithm Description and Overview

For completeness, we restate the essential aspects of the algorithm by Chlamtáč et al. [11]. The algorithm is initially provided a Sparsest-Cut instance $(G, D, \text{cap}, \text{dem})$ alongside a tree decomposition $(\mathcal{T}, \mathcal{B})$ of G. The goal is then to compute a cut in G that has low sparsity.

The algorithm starts by computing, for every vertex set $L = B_i \cup \{s, t\}$ consisting of a bag B_i and a pair of vertices $s, t \in V(G)$, a distribution μ_L over L-assignments. The collection of distributions for all sets L satisfies *consistency* constraints, that is, any two distributions must agree on their joint domains, i.e., $\mu_L|_{L \cap L'} = \mu_{L'}|_{L \cap L'}$ for each pair of sets L, L' with the structure above. Such distributions can be seen as an abstraction of fractional solutions in the context of Sherali-Adams linear programming hierarchies [27], which are convenient in our approach.

If we denote $\text{lpcut}(s, t) = \Pr_{f \sim \mu_{B \cup \{s,t\}}}[f(s) \neq f(t)]$ for any $s, t \in V(G)$, and an arbitrary bag B of $\mathcal{B}$, we can compute the collection of distributions that minimizes

$$\alpha := \frac{\sum_{\{s,t\} \in E_G} \text{cap}_{\{s,t\}} \cdot \text{lpcut}(s, t)}{\sum_{\{s,t\} \in E_D} \text{dem}_{\{s,t\}} \cdot \text{lpcut}(s, t)}.$$

We say that α is the *cut sparsity predicted by distributions* $\{\mu_L\}_L$. Notice that lpcut is well defined by the consistency requirement, since the choice of B does not impact the distribution over $\{s, t\}$-assignments. For ease of notation, we will refer to the implied distribution over some vertex set $X \subseteq B \cup \{s, t\}$ by μ_X, where formally $\mu_X = \mu_{B \cup \{s,t\}}|_X$.

Such a collection of distributions can be computed in time $2^{O(w(\mathcal{T}))} \cdot n^{O(1)}$, using Sherali-Adams linear programming hierarchies restricted to subsets of the form $B_i \cup \{s, t\}$ (see Chlamtáč et al. [11]). This motivates the function name lpcut. Additionally, such a linear program is a relaxation of the Sparsest-Cut problem, and thus α is a lower bound for the minimum sparsity of a cut.

Algorithm 1 obtains a $V(G)$-assignment f by rounding this collection of distributions $\{\mu_L\}$: starting at a bag B_0 (which is considered the root), it chooses an assignment for vertices in B_0 by sampling from the distribution μ_{B_0}; then, for each bag B for which its parent B' is processed, it chooses a B-assignment according to μ_B conditioned on the assignment for $B \cap B'$. As we will see, this rounding approximately preserves the probability of cutting a pair (s, t), thus implying an approximation to the problem.

We now recall a number of useful results about the algorithm and the assignment it computes. Details about the algorithm and the attendant lemmas can be found in the work of Chlamtáč et al. [11].

Denote by $\mathcal{A}$ the distribution over $V(G)$-assignments produced by the algorithm.

Lemma 3.4 ([11, Lemma 3.3]). *For every bag B the assignment $f|_B$ computed by Algorithm 1 is distributed according to μ_B, meaning $\Pr_{f\sim\mathcal{A}}[f|_B = f'] = \Pr_{f^*\sim\mu_B}[f^* = f']$ for all $f' \in \mathcal{F}[B]$.*

A direct consequence of this lemma is the fact that any edge $\{s,t\}$ of G is cut by the algorithm with probability $\mathsf{lpcut}(s,t)$. In particular, the expected capacity of the rounded cut is therefore

$$\sum_{\{s,t\}\in E_G} \mathrm{cap}_{\{s,t\}} \cdot \mathsf{lpcut}(s,t)\,,$$

according to the distributions $\{\mu_L\}$. However, the same property does not hold for the demand edges in D, since the Lemma 3.4 only applies to bags of $\mathcal{T}$. Specifically, the bags of $\mathcal{T}$ do not necessarily contain all of the edges of D, as we do not assume that D has bounded treewidth.

Denote by $\mathsf{algcut}(s,t)$ the probability that the algorithm separates s and t, that is, $\mathsf{algcut}(s,t) = \Pr_{f\sim\mathcal{A}}[f(s) \neq f(t)]$. We would like to lower bound the value $\mathsf{algcut}(s,t) \geq c\,\mathsf{lpcut}(s,t)$ for all demand edges $\{s,t\}$ and some value $c > 0$. This would imply that the expected demand of the rounded cut is at least $c\sum_{\{s,t\}\in E_D} \mathrm{dem}_{\{s,t\}}\,\mathsf{lpcut}(s,t)$, and having a good expected demand and capacity is sufficient for computing a good solution by the following observation.

Observation 3.5 ([11], Remark 4.3). *Let $\{\mu_L\}_L$ be a collection of distributions and α be the cut sparsity predicted by $\{\mu_L\}_L$.*

Then if $\mathsf{algcut}(s,t) \geq c\cdot\mathsf{lpcut}(s,t)$ for all $\{s,t\} \in E_D$ and $\mathsf{algcut}(s,t) = \mathsf{lpcut}(s,t)$ for all $\{s,t\} \in E_G$, we have

$$\mathbb{E}_{f\sim\mathcal{A}}\left[\sum_{\{s,t\}\in E_G} \mathrm{cap}_{\{s,t\}}\,|f(s)-f(t)| - \frac{\alpha}{c}\sum_{\{s,t\}\in E_D} \mathrm{dem}_{\{s,t\}}\,|f(s)-f(t)|\right] \leq 0\,.$$

Furthermore, an assignment f is c-approximate if the value in the expectation above is non-positive, and such a solution can either be obtained by repeated rounding or by derandomization using the method of conditional expectations, without increasing the asymptotic run time.

This observation implies that the bottleneck to obtaining a good approximation factor is the extent to which our rounding algorithm can approximate the marginal of μ_L on the individual edges of D. Our main result relates this marginal to the combinatorial diameter of $\mathcal{T}$. It can now be stated as follows:

Theorem 3.6. *Let $(G, D, \mathrm{cap}, \mathrm{dem})$ be an instance of* Sparsest-Cut*, and $(\mathcal{T},\mathcal{B})$ a tree decomposition of G with width $w(\mathcal{T},\mathcal{B})$ and combinatorial diameter $\Delta(\mathcal{T},\mathcal{B})$.*

Then SC-Round satisfies $\mathsf{algcut}(s,t) \geq \Omega\left(\frac{1}{\Delta(\mathcal{T},\mathcal{B})^2}\right)\cdot\mathsf{lpcut}(s,t)$ for every $\{s,t\} \in E_D$. Thus, we have a factor-$O(\Delta(\mathcal{T},\mathcal{B})^2)$ approximation algorithm for Sparsest-Cut *with run time $2^{O(w(\mathcal{T},\mathcal{B}))}\cdot n^{O(1)}$.*

The rest of this section is devoted to proving this theorem.

3.3 Step 1: Reduction to Short Paths

In this section, we show that when the combinatorial diameter of the tree decomposition is $\delta = \Delta(\mathcal{T},\mathcal{B})$, the analysis can be reduced to the case of a path decomposition of length δ. We employ the following lemma to simplify our analysis of the behavior of the algorithm.

Lemma 3.7 ([11, Lemma 3.4]). *The distribution over the assignments f is invariant under any connected traversal of $\mathcal{T}$, i.e., the order in which bags are processed does not matter, as long as they have a previously processed neighbor. The choice of the first bag B_0 also does not impact the distribution.*

Let $\{s,t\} \in E_D$ be a demand edge. If s and t are contained in a common bag, then $\mathrm{algcut}(s,t) = \mathrm{lpcut}(s,t)$ by Lemma 3.4 and we are done; therefore, we assume that there is no bag containing both s and t. We want to estimate the probability that s and t separated by the algorithm, that is, the probability that $f(s) \neq f(t)$.

The lemma above allows us to reduce to the case in which the algorithm first rounds a bag B_1 containing s, then rounds bags $B_2, \ldots, B_{\ell-1}$ along the path to a bag B_ℓ containing t, and finally B_ℓ. At this point the algorithm has already assigned $f(s)$ and $f(t)$, so the remaining bags of $\mathcal{T}$ can be rounded in any connected order without impacting the separation probability. Hence, it is sufficient to characterize the behavior of the rounding algorithm along paths in $\mathcal{T}$.

Let P be the shortest path connecting a bag containing s to a bag containing t; denote such path by $P = v_1 v_2 \ldots v_\ell$ such that $s \in B_{v_1}$ and $t \in B_{v_\ell}$. By Lemma 3.7, we can assume that the algorithm first processes B_{v_1}, and then all other bags $B_{v_2}, \ldots, B_{v_\ell}$, in this order. Let $\mathcal{B}_P = \{B_{v_i}\}_{i=1}^{\ell}$.

Observe that, except for v_1 and v_ℓ, no other bag of P contains s or t. We repeatedly apply the reduction rule from Definition 3.2 until the resulting path has length at most δ. The following lemma asserts that the distribution of the algorithm is preserved under this reduction rule.

We slightly abuse the notation and denote by $\mathcal{A}$ the distribution of our algorithm on path P starting from v_1.

Lemma 3.8. *Let u, v, w be three consecutive internal nodes on P with $B_v \cap B_w \subseteq B_u$. Let P' be the simplification of P obtained by bypassing v, and let $\mathcal{A}'$ be the distribution obtained by running the algorithm on path P', starting on v_1. Then $\mathcal{A}'$ is exactly the same as $\mathcal{A}$ restricted to $B(P')$.*

Proof. We can assume, without loss of generality, that u, v, w appear on P in the order of rounding; for otherwise, we apply Lemma 3.7 twice: first, to reverse P, and preserve the distribution $\mathcal{A}$; then, to undo the reversing of P' caused by the previous application.

We modify the path decomposition $(P, \mathcal{B}_P)$ into a (tree) decomposition $(\hat{\mathcal{T}}, \hat{\mathcal{B}})$ as follows: remove bag v and add two new bags v', v'' where bag v' is connected to u and w with $B_{v'} = B_u \cap B_v$ and v'' is connected to v' with $B_{v''} = B_v$. This remains a tree decomposition for the vertices in $B(P)$ since vertices in $B_v \setminus B_u$ only occur in the bag $B_{v''}$ (due to our assumption that $B_v \cap B_w \subseteq B_u$).

Running the algorithm SC-Round on $\hat{\mathcal{T}}$ produces exactly the same distribution as $\mathcal{A}$, since we can first round the bags from s to u, then v' and v'', and then the bags from u to t. Since $B(P') = B(P) \setminus (B_{v''} \setminus B_{v'})$, we have that by Lemma 3.7, $\mathcal{A}|_{B(P')}$ is the distribution of SC-Round on the path $\hat{P} = v_1 \ldots u v' w \ldots v_\ell$, obtained by removing v'' from $\hat{\mathcal{T}}$. Now since $B_{v'} \subseteq B_u$, the rounding algorithm in fact does not do anything at bag v', so it can be removed without affecting the distribution. We obtain path P' as a result, and thus $\mathcal{A}|_{B(P')}$ is the same distribution as $\mathcal{A}'$. □

This result allows us conduct the rounding analysis on simplifications of paths. It remains to show that this is beneficial, that is, that the rounding error can be bounded by the length of the path on which we round. As in the work of Chlamtáč et al. [11], we use Markov flow graphs to analyze that error. However, as the length of their paths is $\Theta(\log n)$, they need to derive a bound depending not on that length, but on the width of the decomposition. Since we can now bound the length, we only need a small part of their argument.

3.4 Step 2: Markov Flow Graphs

Let $P = v_1, \ldots, v_\ell$ be a path with length ℓ and $s \in B_{v_1}$, $t \in B_{v_\ell}$. We run Algorithm 1 from v_1 to v_ℓ to compute some assignment f. Let $\mathcal{A}$ be the probability distribution of the resulting assignment f. Recall that $\mathrm{algcut}(s,t)$ denotes the probability that the algorithm assigns $f(s) \neq f(t)$, and $\mathrm{lpcut}(s,t)$ is the probability that s and t are separated according to the distributions $\{\mu_L\}_L$, i.e.,

$\Pr_{f\sim\mu_{B\cup\{s,t\}}}[f(s)\neq f(t)]$. In the second step, we analyze the probability of $\mathsf{algcut}(s,t)$ in terms of $\mathsf{lpcut}(s,t)$. This step is encapsulated in the following lemma.

Lemma 3.9. *There exists a directed layered graph H containing nodes $s_0, s_1, t_0, t_1 \in V(H)$ and a weight function w_H on the edges, satisfying the following properties:*

(1) *For $i = 0, 1$, we have that $\Pr_{f\sim\mathcal{A}}[f(s) = i \ \wedge \ f(t) = 1 - i]$ is at least an $\Omega(1/\ell^2)$-fraction of the minimum (s_i, t_{1-i})-cut of H, where ℓ is the length of the path P.*
(2) *For $i = 0, 1$, the value of a maximum (s_i, t_{1-i})-flow in H is at least $\Pr_{f\sim\mu}[f(s) = i \wedge f(t) = 1-i]$.*

With Lemma 3.9 in hand, we can now complete the proof of Theorem 3.6 as follows.

Proof of Theorem 3.6. We run the algorithm of Chlamtáč et al. to get some $V(G)$-assignment f.

Consider a pair $\{s,t\} \in E_D$. Using Lemma 3.7 and Lemma 3.8, we can reduce the analysis to a path P of length at most $\delta = \Delta(\mathcal{T}, \mathcal{B})$, which is a simplification of a path in $\mathcal{T}$. Now, by Lemma 3.9 applied with $\ell = \delta$ and max-flow-min-cut theorem, we get that

$$\begin{aligned}
\mathsf{algcut}(s,t) &= \Pr_{f\sim\mathcal{A}}[f(s) = 0 \ \wedge \ f(t) = 1] + \Pr_{f\sim\mathcal{A}}[f(s) = 1 \ \wedge \ f(t) = 0] \\
&\geq \Omega\big(1/\delta^2\big)\big(\mathsf{mincut}(s_0,t_1) + \mathsf{mincut}(s_1,t_0)\big) \\
&= \Omega\big(1/\delta^2\big)\big(\mathsf{maxflow}(s_0,t_1) + \mathsf{maxflow}(s_1,t_0)\big) \\
&\geq \Omega\big(1/\delta^2\big)\left(\Pr_{f\sim\mu}[f(s) = 0 \ \wedge \ f(t) = 1] + \Pr_{f\sim\mu}[f(s) = 1 \ \wedge \ f(t) = 0]\right) \\
&= \Omega\big(1/\delta^2\big)\,\mathsf{lpcut}(s,t)\,.
\end{aligned}$$

Recall that, by consistency of the distributions, the probability of an $\{s,t\}$-assignment is the same for every distribution μ_L s.t. $\{s,t\} \subseteq L$; for this reason, we slightly abuse notation and write $\Pr_{f\sim\mu}$ to mean the probability according to any distribution containing $\{s,t\}$.

We conclude that f separates each pair $\{s,t\}$ with probability that is a factor of $O(\delta^2)$ away from $\mathsf{lpcut}(s,t)$ as desired. Applying Observation 3.5 with $c = \Omega(1/\delta^2)$, we can obtain (deterministically) an assignment f^* that is an $O(\delta^2)$-approximation for the Sparsest-Cut instance.

Finally, the run time can be seen to be $2^{O(w(\mathcal{T},\mathcal{B}))} \cdot n^{O(1)}$ using the arguments by Chlamtáč et al. [11], which are summarized as follows: To compute the distributions $\{\mu_L\}_L$, we write a linear program containing a variable for every subset $S \subseteq L$ for every considered set $L = B \cup \{s,t\}$. Therefore, we take subsets of at most $O(n)$ sets, each of size at most $w(\mathcal{T},\mathcal{B}) + 3$, which totals to $O(n) \cdot 2^{w(\mathcal{T},\mathcal{B})}$. Both solving the linear program to obtain the distributions and the rounding procedure are polynomial in the number of variables, thus leading to the claimed run time. □

The rest of this section is dedicated to proving Lemma 3.9. The tools needed for this proof are implicit in the work of Chlamtáč et al. [11]. We restate them for the sake of completeness and in order to adjust it to our terminology. To start, we will take care of a small technical detail inherent to the analysis of Chlamtáč et al., requiring the LP solution to be symmetric in 0 and 1.

Definition 3.10. For any set X and X-assignment g we define the *mirror* of g to be $\overleftrightarrow{g} := \sigma \circ g$, where $\sigma(0) = 1$ and $\sigma(1) = 0$.

Notice that the mirror of an assignment represents the same cut; it merely exchanges the labels. The approximation factor analysis of Chlamtáč et al. requires the distributions to be symmetric in their labeling, in particular $\Pr[f(v) = 0] = \Pr[f(v) = 1]$ for all $v \in V(G)$. They resolve this by demanding symmetry via the Sherali-Adams LP which can be shown to not worsen the relaxation. Using the following lemma, we are able to prove that the rounding analysis also holds in the non-symmetric case.

Lemma 3.11. *Let $G, (\mathcal{T}, \{B_i\}_{i\in V(\mathcal{T})}), \{\mu_L\}$ be the input of Algorithm 1. Consider the modified decomposition $(\mathcal{T}, \{B'_i\}_{i\in V(\mathcal{T}')})$, where a dummy vertex e has been added to every bag B'_i, and the collection of distributions $\{\vec{\mu}_L\}$ defined as:*

$$\vec{\mu}_{L\cup\{e\}}(g', e \to 0) = \frac{1}{2}\mu_L(g'), \quad \vec{\mu}_{L\cup\{e\}}(g', e \to 1) = \frac{1}{2}\mu_L(\overleftrightarrow{g'}) \quad \forall g' \in \mathcal{F}[L]\,.$$

Then $\vec{\mu}$ has the following properties:

(1) *For all $L' = L \cup \{e\}$, $g' \in \mathcal{F}[L']$, we have that $\Pr_{g\sim\vec{\mu}_{L'}}[g = g'] = \Pr_{g\sim\vec{\mu}_{L'}}[g = \overleftrightarrow{g'}]$*
(2) *If g, g^* are random variables representing the resulting assignments of Algorithm 1 run on $G, (\mathcal{T}, \{B_i\}_i), \{\mu_L\}$, and $G, (\mathcal{T}, \{B'_i\}_i), \{\vec{\mu}_{L'}\}$, respectively, then*

$$\Pr[g = g'] + \Pr[g = \overleftrightarrow{g'}] = \Pr[g^*|_{V(G)} = g'] + \Pr[g^*|_{V(G)} = \overleftrightarrow{g'}] \qquad \forall g' \in \mathcal{F}[V(G)]\,.$$

The content of the lemma is at its core not very surprising. If we do not care about the labels, we do not care about whether the algorithm outputs g or $\overleftrightarrow{g}$. But if that is the case, the distributions also should not need to maintain some distinction between the labels. In fact, one could run the algorithm unmodified, and then permute the labels with probability $1/2$. Clearly, this does not change the distribution over cuts, and the choice of labels is now symmetric.

Proof. To make the argument formal, we shall use the value of $g(e)$ to indicate whether or not we are permuting the labels. By Lemma 3.7 we can model the rounding algorithm for $G, (\mathcal{T}, \{B'_i\}_i), \{\vec{\mu}_{L'}\}$ as choosing first a value for $g(e)$, and then proceeding in the same order as the rounding over $G, (\mathcal{T}, \{B_i\}_i), \{\mu_L\}$. With probability $1/2$ we get $g(e) = 0$. Since every bag contains e, e is always conditioned on, so the symmetrized algorithm performs exactly the same computations as the original run would. Meanwhile if $g(e) = 1$, the symmetrized algorithm samples some intermediate assignment g' with exactly the probability that the original algorithm would have sampled $\overleftrightarrow{g'}$.

As a result,

$$\Pr[g^*|_{V(G)} = g'] = \frac{1}{2}\Pr[g = g'] + \frac{1}{2}\Pr[\overleftrightarrow{g} = g'] \qquad \forall g' \in \mathcal{F}[V(G)]\,,$$

which implies

$$\Pr[g^*|_{V(G)} = g'] + \Pr[g^*|_{V(G)} = \overleftrightarrow{g'}] = \Pr[g = g'] + \Pr[g = \overleftrightarrow{g'}] \qquad \forall g' \in \mathcal{F}[V(G)]\,. \qquad \square$$

Notice that while we could construct the symmetrized $\vec{\mu}$ efficiently, we do not need to, as μ is distributed identically, and thus satisfies the same bounds, as $\vec{\mu}$. More concretely, we apply the analysis to $\vec{\mu}$, which is possible because it is symmetric. We can then use Lemma 3.11 to conclude that the probabilities of cutting a pair of vertices in μ and $\vec{\mu}$ are the same, since these probabilities are symmetric, in the sense that $\Pr[f(s) \neq f(t)] = \Pr[f(s) = 1 \wedge f(t) = 0] + \Pr[f(s) = 0 \wedge f(t) = 1]$. Therefore, the bounds on the probabilities of cutting a pair in $\vec{\mu}$ apply also to μ, and thus our results apply obtaining a solution directly from μ.

We now proceed as follows: first, we describe the construction of the graph H, and then we proceed to analyze the values of maximum flow and minimum cut. We will only analyze the flow and cut for $i = 0$, that is, (s_0, t_1)-flow and (s_0, t_1)-cut. The other case is analogous.

Construction of Graph H: By Lemma 3.11 we can assume that the distributions $\{\mu_L\}_L$ are symmetric in the labels $\{0,1\}$. In particular, this gives $\Pr[f(v)=1]=\Pr[f(v)=0]=1/2$ for any vertex v.

The rounding can be modeled by a simple Markov process. Denote by $I_0,\ldots,I_\ell$ the sets that are conditioned on in Algorithm 1, $I_i = B_{v_i}\cap B_{v_{i+1}}$ for $i\in\{1,\ldots,\ell-1\}$; we refer to these sets as *conditioning sets*.

For the initial and final sets of the rounding procedure we take $I_0=\{s\}$, $I_\ell=\{t\}$. Now we are ready to describe our graph H:

- **Vertices**: Vertices of H are arranged into layers $L_0,\ldots,L_\ell$ with $L_i=\mathcal{F}[I_i]$. Observe that $|L_i|=2^{|I_i|}$. The vertices of H represent the intermediate states the algorithm might reach.
- **Edges**: For each i, there is a directed edge from every vertex in L_i to every vertex in L_{i+1}. The weight of the edge (f_i,f_{i+1}), for $f_i\in L_i$, $f_{i+1}\in L_{i+1}$, is equal to the probability of the joint event, $w_H(f_i,f_{i+1})=\Pr[f|_{I_i}=f_i\wedge f|_{I_{i+1}}=f_{i+1}]$.
  We remark that the weight is 0 whenever f_i and f_{i+1} are contradictory, and that probabilities are well defined, as $I_i\cup I_{i+1}\subseteq B_{i+1}$.

Observe that the weight of an edge is the probability that both of its endpoints are reached by the algorithm, and hence the probability that the algorithm transitions along that edge.

Observation 3.12. *Let $\mathcal{I}=\bigcup_i I_i$. The distribution $\mathcal{A}|_{\mathcal{I}}$ can be viewed as the following random walk in H: Pick a random vertex in L_0 and start taking a random walk where each edge is taken with probability proportional to its weight. Formally, once a node f_i is reached, choose the next node f_{i+1} with probability $w_H(f_i,f_{i+1})/\Pr[f|_{I_i}=f_i]$.*

At this point, we rename $\mathcal{A}:=\mathcal{A}|_{\mathcal{I}}$. Notice that the layer L_0 contains two vertices corresponding to the assignment $f(s)=0$ and $f(s)=1$, respectively. We denote them by $L_0=\{s_0,s_1\}$. Similarly, $L_\ell=\{t_0,t_1\}$. Notice further that $\Pr_{f\sim\mathcal{A}}[f(s)=0,f(t)=1]$ is exactly the probability that the random walk starts at $s_0\in L_0$ and ends at $t_1\in L_\ell$.

Maximum (s_0,t_1)-Flow: We are now ready to show that the value of the maximum (s_0,t_1)-flow is at least $\Pr_{f\sim\mu}[f(s)=0,f(t)=1]$.

We define the flow $g\colon E(H)\to\mathbb{R}_{\geq 0}$ as follows, for $i\in\{1,\ldots,\ell-1\}$, $f_i\in L_i$ and $f_{i+1}\in L_{i+1}$:

$$g(f_i,f_{i+1})=\Pr_{f\sim\mu_{B_{v_{i+1}}\cup\{s,t\}}}[f(s)=0,f(t)=1,f|_{I_i}=f_i,f|_{I_{i+1}}=f_{i+1}]\,.$$

We remark that g is an s_0-t_1-flow, that is, it satisfies flow conservation at all vertices in H except s_0,t_1, and the capacities of graph H are respected, that is, $g(e)\leq w_H(e)$ for all $e\in E(H)$. The value of g is given by:

$$\begin{aligned}\sum_{(s_0,f^*)\in\delta^+(s_0)} g(s_0,f^*)&=\sum_{f^*\in\mathcal{F}[I_1]}\Pr_{f\sim\mu_{B_{v_1}\cup\{s,t\}}}[f(s)=0,f(t)=1,f|_{I_1}=f^*]\\&=\Pr_{f\sim\mu_{B_{v_1}\cup\{s,t\}}}[f(s)=0,f(t)=1]\,.\end{aligned}$$

This concludes the proof of Point 2 of Lemma 3.9.

A Potential Function: Before we show a cut with the desired capacity, we need to introduce some notation. For $i=0,\ldots,\ell$, let X_i be a random variable indicating the vertex in L_i visited by the random walk (i.e., picked by the algorithm. We denote by $\mathbf{X}=X_0X_1\ldots X_\ell$ the path taken in the random walk process. We can interchangeably view distribution $\mathcal{A}$ as either the distribution that samples an assignment $f\colon\mathcal{I}\to\{0,1\}$ or one that samples a (random walk) path $\mathbf{X}$.

We define, for every layer L_i and every vertex $v \in L_i$,

$$A(v) := \Pr_{\mathbf{X}\sim\mathcal{A}}[X_0 = s_0 \mid X_i = v] - \frac{1}{2}\,.$$

Intuitively, this function captures the extent to which v has information about the initial state of the Markov process. On the one hand, if $A(v)$ is equal to 0, v knows essentially nothing about X_0, the choice of v does not imply anything about X_0. On the other hand, if $A(v)$ is far from 0, then we can glean some information about X_0 from v being visited; in particular, if the probability that s and t are cut is low, we must have $A(t_1) \approx -1/2$.

To track how A changes from layer to layer, we use the potential function $\phi\colon \{0,\dots,\ell\} \to \mathbb{R}_{\geq 0}$, which we define as:

$$\phi(i) := \mathrm{Var}_{\mathbf{X}\sim\mathcal{A}}[A(X_i)]\,.$$

The following argument by Chlamtáč et al. bounds the change in potential in terms of the probability that $X_0 = s_0$ and $X_\ell = t_1$.

Corollary 3.13 (From [11, Lemma 5.2]). *Without loss of generality, we can assume that* $\phi(0) - \phi(\ell) \leq 2\Pr_{\mathbf{X}\sim\mathcal{A}}[X_0 = s_0 \wedge X_\ell = t_1]$.

Proof. Suppose that $\Pr_{\mathbf{X}\sim\mathcal{A}}[X_0 = s_0 \wedge X_\ell = t_1] > \frac{1}{4}$. Then we already have that $\mathrm{algcut}(s,t) \geq \frac{1}{2}\,\mathrm{lpcut}(s,t)$ and thus do not need any further analysis. Otherwise, we observe that L_0 and L_ℓ only contain two nodes each. We can therefore compute the variances explicitly:

$$\begin{aligned}
\phi(0) - \phi(\ell) &= A(s_0)^2 - A(t_1)^2 \\
&= \frac{1}{4} - A(t_1)^2 \\
&= \frac{1}{4} - (2\Pr_{\mathbf{X}\sim\mathcal{A}}[X_0 = s_0 \wedge X_\ell = t_1] - \frac{1}{2})^2 \\
&\leq 2\Pr_{\mathbf{X}\sim\mathcal{A}}[X_0 = s_0 \wedge X_\ell = t_1]\,.
\end{aligned}$$

□

Minimum (s_0,t_1)-cut: We are now ready to analyze the value of minimum (s_0,t_1)-cut in H. It suffices to give a lower bound on $\phi(0)-\phi(\ell)$ by Corollary 3.13. The following lemma from Chlamtáč et al. [11] is necessary to control the change of ϕ when moving between layers.

Lemma 3.14 ([11, Lemma 5.2]). *For all $0 < l_1 < l_2 < \ell$ we have*

$$\phi(l_1) - \phi(l_2) = \sum_{u\in L_{l_1}, v\in L_{l_2}} \Pr_{\mathbf{X}\sim\mathcal{A}}[X_{l_1} = u \wedge X_{l_2} = v](A(u) - A(v))^2\,.$$

With Lemma 3.14 it is possible to prove the following lemma which is shown implicitly by Chlamtáč et al. [11]. We provide a full proof for completeness.

Lemma 3.15 (Analogous to [11, Lemma 5.4]). *Let C be the set of edges (f_i, f_{i+1}) in $E(H)$ such that $|A(f_i) - A(f_{i+1})| \geq \rho$, for some threshold $\rho > 0$. Then $\sum_{e\in C} w_H(e) \leq (\phi(0) - \phi(\ell)) \cdot 1/\rho^2$.*

Proof. It holds that

$$\begin{aligned}\sum_{e\in C} w_H(e) &= \sum_{i\in 0,\dots,\ell-1} \sum_{\substack{f_i\in L_i, f_{i+1}\in L_{i+1}\\ |A(f_i)-A(f_{i+1})|\geq\rho}} w_H(f_i,f_{i+1})\\ &\leq \frac{1}{\rho^2}\sum_{i\in 0,\dots,\ell-1} \sum_{\substack{f_i\in L_i, f_{i+1}\in L_{i+1}\\ |A(f_i)-A(f_{i+1})|\geq\rho}} w_H(f_i,f_{i+1})(A(f_i)-A(f_{i+1}))^2\\ &\leq \frac{1}{\rho^2}\sum_{i\in 0,\dots,\ell-1}(\phi(i+1)-\phi(i))\\ &= \frac{1}{\rho^2}(\phi(0)-\phi(\ell)),\end{aligned}$$

where the second inequality is an application of Lemma 3.14. □

We can apply Lemma 3.15 in the following fashion. Suppose $A(t_1) \geq 0$. In that case we have $\Pr[X_0 = s_0 \mid X_\ell = t_1] \geq 1/2$, so s and t are cut with probability at least $\frac{1}{2}\operatorname{lpcut}(s,t)$. This error is already small enough, so assume $A(t_1) < 0$. Then $A(s_0) - A(t_1) > 1/2$. Since every path from s_0 to t_1 has exactly ℓ edges, any such path must contain an edge (f_i, f_j) with $A(f_i) - A(f_j) > 1/(2\ell)$. Cutting all such edges therefore separates s_0 and t_1. Hence, by applying Lemma 3.15, the minimum s_0-t_1-cut has size at most

$$\begin{aligned}O(\ell^2)(\phi(0)-\phi(\ell)) &\leq O(\ell^2)\Pr[X_0 = s_0 \wedge X_\ell = t_1]\\ &= O(\ell^2)\Pr[f(s) = 0 \wedge f(t) = 1]\,.\end{aligned}$$

This concludes the proof of Point 1 of Lemma 3.9. We see that the cutting probability predicted by the distributions is realised by the rounded solution f, up to a factor $\Omega(1/\ell^2)$.

The analysis of this section gives an alternative to the one given by Chlamtáč et al.. Their constant of approximation depends on the size of the layers (i.e., the width of the bags of the tree decomposition) of H rather than the number of layers (i.e., the length of the path on which the analysis is performed). While the layer sizes depend only on k, the dependence is exponential. The number of layers meanwhile is a priori $\log(n)$, which would give a worse approximation guarantee. However, we will show how to modify a tree decomposition to ensure that all paths used in the analysis are short so that H has few layers.

4 COMBINATORIALLY SHALLOW TREE DECOMPOSITIONS

In this section, we show how to construct tree decompositions with low combinatorial diameter, thus achieving the approximation results stated in Theorem 1.1. We start by restricting our consideration to decompositions that are shallow in the traditional sense. For a given graph G with treewidth k, we consider a tree decomposition $(\mathcal{T}, \mathcal{B})$ with diameter $d = O(\log n)$ and width $O(k)$ [4, 20] (see Section 2 for details). Fix some root r in $V(\mathcal{T})$.

Our goal is now to modify $(\mathcal{T}, \mathcal{B})$ such that every node has a combinatorially short path to r. This is a necessary requirement, but perhaps surprisingly it is not sufficient. The combinatorial lengths of paths do not necessarily induce a metric[3] on $V(\mathcal{T})$, and therefore bounding the length to r does not on its own suffice to bound the combinatorial diameter. Instead, for any pair $s, t \in V(\mathcal{T})$, we will bound the combinatorial length from s and t to their lowest common ancestor, which is sufficient to bound the combinatorial length of $\mathcal{T}_{s\leftrightarrow t}$.

[3]Consider bags $\{ab\}, \{abc\}, \{acd\}, \{ade\}, \{aef\}, \{afg\}, \{a\}$ occuring in that order as a path. The whole path can be reduced to just the endpoints. The subpath $\{ab\}, \{abc\}, \{acd\}, \{ade\}, \{aef\}, \{afg\}$ is irreducible. Thus the distance from $\{ab\}$ to $\{afg\}$ is larger than the sum of the distances from $\{ab\}$ to $\{a\}$ and $\{afg\}$ to $\{a\}$.

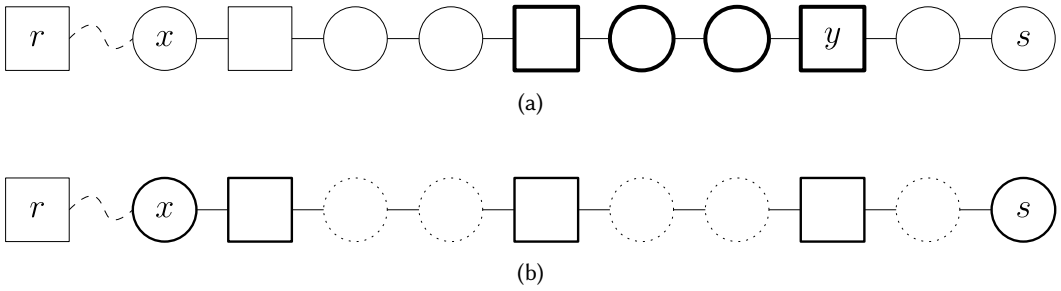

Fig. 2. Illustration of a path from the root to some node s. The square nodes are the synchronization nodes. The bridge from y to its synchronization ancestor is marked in bold in Figure 2a. The dotted nodes in Figure 2b mark those nodes which can be removed when simplifying the x-s-path in $\mathcal{T}'$.

We will not show explicitly that the modified structures are in fact tree decompositions. However, the following lemma can be used to verify this fact for all of our constructions.

Lemma 4.1. *Let $(\mathcal{T}, \{B_i\}_{i\in V(\mathcal{T})})$ be a tree decomposition of a graph G, rooted at r. Then $(\mathcal{T}, \{B'_i\}_{i\in V(\mathcal{T})})$ is also a tree decomposition of G if $B_i \subseteq B'_i \subseteq B_i \cup B'_{p(i)}$ for all bags B_i.*

Proof. Fix some $s \in V(G)$. We need to show that $\mathcal{T}'_s := \{i \in V(\mathcal{T}) \mid s \in B'_i\}$ is connected. As $\mathcal{T}_s := \{i \in V(\mathcal{T}) | s \in B_i\}$ is connected and $\mathcal{T}_s \subseteq \mathcal{T}'_s$, it suffices to show that any $i \in \mathcal{T}'_s$ is connected to $\mathcal{T}_s$ in $\mathcal{T}'_s$. We do this by induction over the distance of i to the root.

For $i = r$ we have $B'_r = B_r$, so either $i \notin \mathcal{T}'_s$ or $i \in \mathcal{T}_s$. Otherwise, consider some $i \in \mathcal{T}'_s$, so $s \in B_i \cup B'_{p(i)}$. Then we either have $s \in B_i$, in which case we are done, or $s \in B'_{p(i)}$. But this gives $p(i) \in \mathcal{T}'_s$, and $p(i)$ is closer to r than i. Thus we can assume that $p(i)$ is connected to $\mathcal{T}_s$, and hence i is also connected to $\mathcal{T}_s$ via $p(i)$. □

We introduce three objects, which we call **bridges**, **highways**, and **super-highways**, and show that they can be used to prove the three parts of Theorem 1.1.

4.1 Bridges

Fix a parameter $\lambda \in \{1, \dots, d\}$. Define $\ell\colon V(\mathcal{T}) \to \mathbb{N}_0$ to be the *level* of a node in $\mathcal{T}$, that is, $\ell(i)$ is the number of edges on $\mathcal{T}_{i\leftrightarrow r}$.

Definition 4.2. We call a node $i \in V(\mathcal{T})$ a *synchronization node* if its level is a multiple of λ. Define also the *synchronization ancestor* $\sigma(v)$ of any node v to be the first node on the path from v to r that is a synchronization node, excluding v itself.

We can construct a tree decomposition $(\mathcal{T}', \mathcal{B}' = \{B'_i\}_{i\in V(\mathcal{T}')})$ by taking $\mathcal{T}' = \mathcal{T}$ and setting $B'_i = B(\mathcal{T}_{i\leftrightarrow\sigma(i)})$ for each node $i \in V(\mathcal{T})$, that is, the new bag is obtained by combining all the bags from v up to its synchronization ancestor. This increases the width of the decomposition by a factor of at most λ. We may view this path connecting v to the synchronization point as a **bridge** crossing over all intermediate nodes in one step.

Lemma 4.3. *It holds that $(\mathcal{T}', \mathcal{B}')$ has combinatorial diameter $O(d/\lambda)$.*

Proof. Fix any two nodes $s, t \in V(\mathcal{T}')$ and take x to be their lowest common ancestor in $\mathcal{T}'$. Then the combinatorial length of $\mathcal{T}'_{s\leftrightarrow t}$ is at most the sum of the combinatorial lengths of $\mathcal{T}'_{s\leftrightarrow x}$ and $\mathcal{T}'_{x\leftrightarrow t}$.We remark that triangle inequality holds in this case, because x is on the path from s to t. Thus, it suffices to show that the combinatorial length of $\mathcal{T}'_{s\leftrightarrow x}$ is $O(d/\lambda)$. The result follows analogously for $\mathcal{T}'_{x\leftrightarrow t}$.

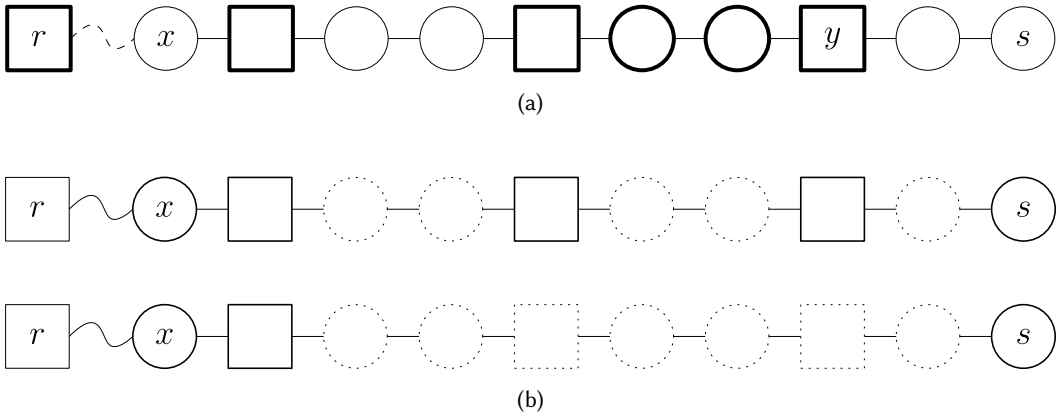

Fig. 3. The bold nodes in Figure 3a mark the bridge and highway from y to r. The dotted nodes in Figure 3b illustrate the redundant nodes during the two simplification rounds for the x-s-path, leaving a path of length 2.

Using the rules of Definition 3.2, we can bypass any node that is neither a synchronization node nor s or x. To do this, iteratively pick any such node v with neighbors u and w, where u is the neighbour closer to s. The order in which these nodes v are bypassed does not matter. Because no synchronization nodes are ever bypassed, u must be somewhere on the subpath from v to the first synchronization node below v. Then we have $\sigma(v) = \sigma(u)$, giving $B'_v \subseteq B'_u$ and in particular $B'_v \cap B'_w \subseteq B'_u$ so that v can be bypassed.

Therefore, the path $\{v \in \mathcal{T}'_{s\leftrightarrow x} | v = s \vee v = x \vee v \text{ is a synchronization node}\}$ is a simplification of $\mathcal{T}'_{s\leftrightarrow x}$. Since there are at most d/λ synchronization nodes on any upward path, the lemma follows. □

This lemma, in conjunction with Theorem 3.6 and the fact that $(\mathcal{T}', \mathcal{B}')$ can be computed in polynomial time from $(\mathcal{T}, \mathcal{B})$, yields:

Corollary 4.4. *For every λ, there is an algorithm that computes an $O((d/\lambda)^2)$-approximation for* Sparsest-Cut *instances where G has treewidth at most k, in time $2^{O(\lambda k)} \cdot n^{O(1)}$.*

Setting $\lambda = \lceil d/k \rceil$ results in an $O(k^2)$-approximation in time $2^k \cdot n^{O(1)}$, while setting $\lambda = d$ gives an $O(1)$-approximation in time $n^{O(k)}$.

4.2 Highways

The idea of extending bags towards the root can be exploited further by adding the vertices in a synchronization bag to all of its descendants. We may regard this as giving each node a bridge to the next synchronization node, as well as a **highway** along the synchronization nodes towards the root. This idea leads to the following construction.

Let $(\mathcal{T}', \{B'_i\}_{i \in V(\mathcal{T}')})$ be a modified tree decomposition with $\mathcal{T}' = \mathcal{T}$ as before, and

$$B'_v := B(\{w \in \mathcal{T}_{v\leftrightarrow r} \mid w \in \mathcal{T}_{v\leftrightarrow\sigma(v)} \vee w \text{ is a synchronization node}\}) .$$

for each $v \in V(\mathcal{T})$.

The size of these bags is at most $(k+1)(\lambda+d/\lambda)$, which for $\lambda = d/k$ is in $O(d+k^2) \subseteq O(\log n + k^2)$.

Notice that the bag B_r is now contained in any bag B'_i, so we have some hope that the combinatorial diameter of $(\mathcal{T}', \{B'_i\}_{i \in V(\mathcal{T}')})$ is low. Indeed this is true.

Lemma 4.5. *It holds that $\mathcal{T}'$ has combinatorial diameter at most* 3.

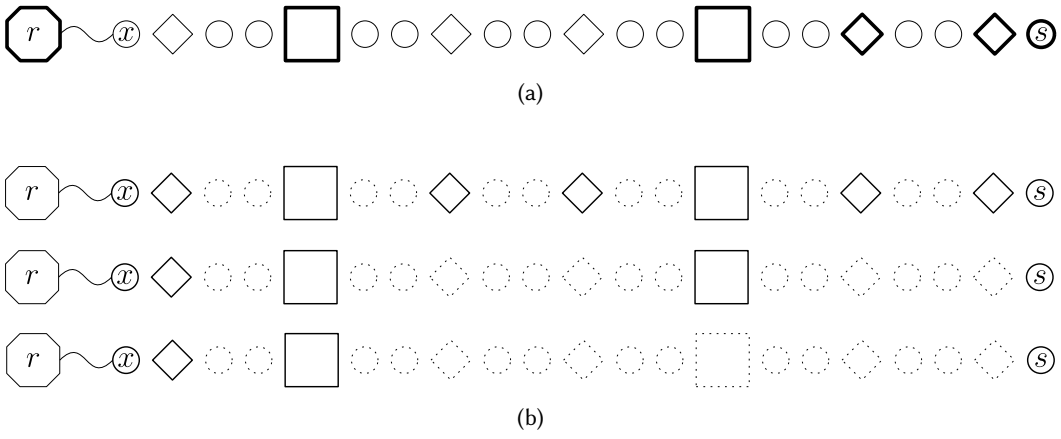

Fig. 4. Illustration of an upward path with nodes of layer −1 as circles, nodes of layer 0 as diamonds, and nodes of layer 1 as squares. The root is at some unspecified maximum layer. The bold nodes inFigure 4a mark the super-highway from s to r. Figure 4b illustrates the simplification rounds for the x-s-path, removing all nodes of some layer in each round, except x, s, and possibly one node close to x.

Proof. As before, we split any s-t-path at x, the lowest common ancestor of s and t, and consider only the s-x-path. Every non-synchronization node v on $\mathcal{T}_{s\leftrightarrow x}$ has a node below it which is either a synchronization node or s. The bag of that node is a superset of B'_v, so all non-synchronization nodes except s and x can be bypassed. Call that reduced path P.

Consider the neighbor of s in P, which we denote v, and assume that v is not the neighbor of x in P. Then v must be a synchronization node, and its next node in P is $\sigma(v)$. Now, the intersection $B'_v \cap B'_{\sigma(v)}$ contains exactly all of the bags of synchronization nodes in $\mathcal{T}_{\sigma(v)\leftrightarrow r}$, and thus, $B'_v \cap B'_{\sigma(v)} \subseteq B'_s$. This implies that v can be bypassed, and by repeating this process, we can bypass every synchronization node except for the neighbor of x.

This gives a possible simplification of $\mathcal{T}_{s\leftrightarrow t}$ as the path $(s, \sigma_s, x, \sigma_t, t)$, where the σ_s and σ_t are the synchronization nodes below x on the paths to s and t, respectively. There is a further reduction of the whole path, since B'_x is precisely $B'_{\sigma_s} \cap B'_{\sigma_t}$. This allows us to remove x as well, giving a simplification of length 3. □

Using the fact that $d \in O(\log n)$, and setting $\lambda = d/k$ gives a fixed-parameter algorithm that yields a constant-factor approximation:

Corollary 4.6. *There exists an algorithm that in time $2^{O(k^2)} \cdot n^{O(1)}$ computes a factor-$O(1)$ approximation for* Sparsest-Cut *instances where G has treewidth at most k.*

4.3 Super-Highways

We can think of the previous construction as having two layers, bridges to synchronization nodes and highways along synchronization nodes to the root. The highways need to cover many synchronization nodes, leading to large bags in $\mathcal{T}'$. To improve on this we introduce a network of **super-highways** of different layers, where each layer covers fewer, more spaced-out synchronization nodes on a root-leaf path. When we connect a node to the root we can then move up the tree layer by layer with increasing speed, decreasing the size of bags in $\mathcal{T}'$. Each added layer is paid for by the need for an additional node in path simplifications for moving between layers, giving a trade-off between run time and approximation guarantee.

Let $q \in \mathbb{N}_1$ be a parameter representing the number of layers. For a node $v \in \mathcal{T}$, we define the *layer* $\pi(v)$ of v as

$$\pi(v) := \max\{-1, \max\{j \in \{0, \dots, q-1\} \mid \ell(v) \equiv 0 \mod k^{j/q} d/k\}\}.$$

By this definition, all synchronization nodes are assigned to some non-negative layer, and all other nodes are on layer -1. We now get a new tree decomposition $(\mathcal{T}', \{B'_i\}_{i \in V(\mathcal{T}')})$ by constructing bags

$$B'_v = B(\{w \in \mathcal{T}_{p(v) \leftrightarrow r} \mid \pi(w) = \max\{\pi(u) \mid u \in \mathcal{T}_{p(v) \leftrightarrow w}\}\} \cup \{v\})$$

for each $v \in V(\mathcal{T}')$. Informally, we start at some node v and move towards r by first taking all nodes of layer -1 until we hit a node of layer 0, then taking only nodes of layer 0 until we hit layer 1, and so on. The nodes at higher layers are spaced further apart. Thus this process "speeds up" thereby generating smaller bags. To be precise, there are q layers and at most $k^{1/q}$ nodes of any one layer in a bag, so $(\mathcal{T}', \mathcal{B}')$ has width $O(d + qk^{1+1/q})$.

We now show that $(\mathcal{T}', \{B'_i\}_{i \in V(\mathcal{T})})$ has combinatorial diameter depending only on q.

Lemma 4.7. *It holds that $(\mathcal{T}', \{B'_i\}_{i \in V(\mathcal{T}')})$ has combinatorial diameter at most $2q + 1$.*

Proof. As before, we only show that any upward path from s to x has combinatorial length at most $q + 1$. We need to perform a round of reductions for every layer, with the goal of leaving only s, x, as well as the first node of at least that layer below x. For layer -1, this holds with the same argument as before.

We can now proceed by induction, fixing some layer i and assuming that the s-x-path P has been reduced to contain only s, then nodes of layers $\geq i$, followed by a sequence $(\sigma_{i-1}, \sigma_{i-2}, \dots, \sigma_0, x)$, where each node σ_j is in layer j. Here, we assume w.l.o.g. that x is at layer -1. Now consider any node v of layer i, except the one closest to x. Because its neighbors also have level at least i (or are s), the intersection of their bags can be represented as the union of bags of $\mathcal{T}$ whose layer is at least i. Let w be the predecessor of v on s-x-path P. The set B'_w is constructed from some upward path starting at w, containing only nodes of non-decreasing layer. This upward path hits layer i between w and v, but not layer $i + 1$ since a node of layer $i + 1$ would be on P between w and v. So then B'_w covers all nodes of layer at least i that B'_v covers, and therefore v can be bypassed, concluding the inductive step.

The simplification of $\mathcal{T}_{s \leftrightarrow x}$ produced in this fashion is a path $(s, \sigma_{q-1}, \dots, \sigma_0, x)$, where $\pi(\sigma_i) = i$. If we add the same simplification for $\mathcal{T}_{t \leftrightarrow x}$ we get a simplification for $\mathcal{T}_{s \leftrightarrow t}$ that takes the form $(s, \sigma_{q-1}, \dots, \sigma_0, x, \sigma'_0, \dots, \sigma'_{q-1}, t)$. As before x can be bypassed since its bag is the intersection of the bags of σ_0 and σ'_0. Thus any s-t-path in $\mathcal{T}'$ has combinatorial length at most $2q + 1$. □

This implies the existence of the following family of algorithms.

Corollary 4.8. *There exists an algorithm that, for any $q \in \mathbb{N}_1$, computes a factor-$O(q^2)$ approximation for* Sparsest-Cut *on graphs of treewidth k in time $O(2^{qk^{1+1/q}}) \cdot n^{O(1)}$. In particular, taking $q = \log k$ gives a factor-$O(\log^2 k)$ approximation in time $2^{O(k \log k)} \cdot n^{O(1)}$.*

5 CONCLUSION & OPEN PROBLEMS

Our work is an attempt to simultaneously obtain the best run time and approximation factor for Sparsest-Cut in the low-treewidth regime. Our research question combines the flavors of two very active research areas, namely parameterized complexity and approximation algorithms. We introduce a new measure of tree decomposition called combinatorial diameter and show various constructions with different tradeoffs between $w(\mathcal{T}, \mathcal{B})$ and $\Delta(\mathcal{T}, \mathcal{B})$. We leave the question of getting 2-approximation in $2^{O(k)} \cdot n^{O(1)}$ time as the main open problem. One way to design such

an algorithm is to show the existence of a tree decomposition with $w(\mathcal{T}, \mathcal{B}) = O(\log n + k)$ and $\Delta(\mathcal{T}, \mathcal{B}) = 2$.

Another interesting question is to focus on polynomial-time algorithms and optimize the approximation factor with respect to treewidth. In particular, is there an $\log^{O(1)} k$ approximation in polynomial time? This question is open even for the *uniform* Sparsest-Cut (unit demand for every vertex pair), for which a fixed-parameter algorithm [5] but no polynomial-time algorithm is known.

A broader direction that would perhaps complement the study along these lines is to improve our understanding on a natural LP-rounding algorithm on the lift-and-project convex programs in general. For instance, can we prove a similar tradeoff result for other combinatorial optimization problems in this setting? One candidate problem is the Group Steiner Tree problem, for which a factor-$O(\log^2 n)$ approximation in time $n^{O(k)}$ is known (and the algorithm there is "the same" algorithm as used for finding sparsest cuts). Can we get a factor-$O(\log^2 n)$ approximation in time $2^{O(k)} \cdot n^{O(1)}$?

ACKNOWLEDGMENTS

Parinya Chalermsook has been supported by European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No. 759557) and by Academy of Finland Research Fellowship, under grant number 310415. Joachim Spoerhase has been partially supported by European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No. 759557). Daniel Vaz has been supported by the grant ANR-19-CE48-0016 from the French National Research Agency (ANR), as well as the Alexander von Humboldt Foundation with funds from the German Federal Ministry of Education and Research (BMBF); the work was partially completed while at Technische Universität München.

REFERENCES

[1] Sanjeev Arora, James Lee, and Assaf Naor. 2008. Euclidean distortion and the sparsest cut. *J. American Math. Soc.* 21, 1 (2008), 1–21.

[2] Sanjeev Arora, Satish Rao, and Umesh Vazirani. 2009. Expander flows, geometric embeddings and graph partitioning. *J. ACM* 56, 2 (2009), 5.

[3] Daniel Bienstock and Nuri Ozbay. 2004. Tree-width and the Sherali-Adams operator. *Discrete Optim.* 1, 1 (2004), 13–21.

[4] Hans L Bodlaender. 1988. NC-Algorithms for graphs with small treewidth. In *Proc. WG 1988 (Lecture Notes Comput. Sci., Vol. 344)*. 1–10.

[5] Paul Bonsma, Hajo Broersma, Viresh Patel, and Artem Pyatkin. 2012. The complexity of finding uniform sparsest cuts in various graph classes. *J. Discrete Algorithms* 14 (2012), 136–149.

[6] Parinya Chalermsook, Syamantak Das, Guy Even, Bundit Laekhanukit, and Daniel Vaz. 2018. Survivable network design for group connectivity in low-treewidth graphs. In *Proc. APPROX-RANDOM 2018 (Leibniz Int. Proc. Informatics, Vol. 116)*. 8:1–8:19.

[7] Parinya Chalermsook, Syamantak Das, Bundit Laekhanukit, and Daniel Vaz. 2017. Beyond metric embedding: Approximating group Steiner trees on bounded treewidth graphs. In *Proc. SODA 2017*. 737–751.

[8] Shuchi Chawla, Robert Krauthgamer, Ravi Kumar, Yuval Rabani, and D. Sivakumar. 2006. On the hardness of approximating multicut and sparsest-cut. *computational complexity* 15, 2 (2006), 94–114.

[9] Chandra Chekuri, Anupam Gupta, Ilan Newman, Yuri Rabinovich, and Alistair Sinclair. 2006. Embedding k-outerplanar graphs into ℓ_1. *SIAM J. Discrete Math.* 20, 1 (2006), 119–136.

[10] Eden Chlamtáč, Robert Krauthgamer, and Prasad Raghavendra. 2010. Approximating sparsest cut in graphs of bounded treewidth. In *Proc. APPROX/RANDOM 2010 (Lecture Notes Comput. Sci., Vol. 6302)*. 124–137.

[11] Eden Chlamtáč, Robert Krauthgamer, and Prasad Raghavendra. 2010. *Approximating sparsest cut in graphs of bounded treewidth.* Technical Report. https://arxiv.org/abs/1006.3970.

[12] Julia Chuzhoy and Sanjeev Khanna. 2009. Polynomial flow-cut gaps and hardness of directed cut problems. *J. ACM* 56, 2 (2009), 1–28.

[13] Vincent Cohen-Addad, Anupam Gupta, Philip N. Klein, and Jason Li. 2021. A quasipolynomial ($2+\varepsilon$)-approximation for planar sparsest cut. In *Proc. STOC 2021*. 1056–1069.

[14] Vincent Cohen-Addad, Tobias Mömke, and Victor Verdugo. 2021. *A 2-approximation for the bounded treewidth sparsest cut problem in FPT time.* Technical Report. https://arxiv.org/abs/2111.06163.
[15] Vincent Cohen-Addad, Tobias Mömke, and Victor Verdugo. 2022. A 2-approximation for the bounded treewidth sparsest cut Problem in FPT time. In *Proc. IPCO 2022 (Lecture Notes Comput. Sci., Vol. 13265).* 112–125.
[16] Naveen Garg, Goran Konjevod, and Ramamoorthi Ravi. 2000. A polylogarithmic approximation algorithm for the group Steiner tree problem. *J. Algorithms* 37, 1 (2000), 66–84.
[17] Anupam Gupta, Ilan Newman, Yuri Rabinovich, and Alistair Sinclair. 2004. Cuts, trees and ℓ_1-embeddings of graphs. *Combinatorica* 24, 2 (2004), 233–269.
[18] Anupam Gupta, Kunal Talwar, and David Witmer. 2013. Sparsest cut on bounded treewidth graphs: Algorithms and hardness results. In *Proc. STOC 2013.* 281–290.
[19] Eran Halperin and Robert Krauthgamer. 2003. Polylogarithmic inapproximability. In *Proc. STOC 2003.* 585–594.
[20] Tuukka Korhonen. 2022. A single-exponential time 2-approximation algorithm for treewidth. In *Proc. FOCS 2021.* 184–192.
[21] Monique Laurent. 2003. A comparison of the Sherali-Adams, Lovász-Schrijver, and Lasserre relaxations for 0-1 programming. *Math. Oper. Res.* 28, 3 (2003), 470–496.
[22] Anupam Gupta (lecturer) and Amitabh Basu (scribe). 2008. Lecture 19: Sparsest Cut and ℓ_1 Embeddings. https://www.cs.cmu.edu/~anupamg/adv-approx/lecture19.pdf. [Online; accessed 2021-09-26].
[23] James R. Lee and Anastasios Sidiropoulos. 2013. Pathwidth, trees, and random embeddings. *Combinatorica* 33, 3 (2013), 349–374.
[24] Tom Leighton and Satish Rao. 1999. Multicommodity max-flow min-cut theorems and their use in designing approximation algorithms. *J. ACM* 46, 6 (1999), 787–832.
[25] Avner Magen and Mohammad Moharrami. 2009. Robust algorithms for on minor-free graphs based on the Sherali-Adams hierarchy. In *Proc. APPROX/RANDOM 2009 (Lecture Notes Comput. Sci., Vol. 5687).* 258–271.
[26] David W. Matula and Farhad Shahrokhi. 1990. Sparsest cuts and bottlenecks in graphs. *Discret. Appl. Math.* 27, 1-2 (1990), 113–123.
[27] Hanif D Sherali and Warren P Adams. 1990. A hierarchy of relaxations between the continuous and convex hull representations for zero-one programming problems. *SIAM J. Discrete Math.* 3, 3 (1990), 411–430.
[28] Martin J. Wainwright and Michael I. Jordan. 2004. *Treewidth-based conditions for exactness of the Sherali-Adams and Lasserre relaxations.* Technical Report. Technical Report 671, University of California, Berkeley.