On the advantage of overlapping clusters for minimizing conductance

Rohit Khandekar* Guy Kortsarz† Vahab Mirrokni‡

January 9, 2013

Abstract

Graph clustering is an important problem with applications to bioinformatics, community discovery in social networks, distributed computing, and more. While most of the research in this area has focused on clustering using *disjoint* clusters, many real datasets have *inherently overlapping* clusters. We compare overlapping and non-overlapping clusterings in graphs in the context of minimizing their conductance. It is known that allowing clusters to overlap gives better results in practice. We prove that overlapping clustering may be significantly better than non-overlapping clustering with respect to conductance, even in a theoretical setting.

For minimizing the *maximum* conductance over the clusters, we give examples demonstrating that allowing overlaps can yield significantly better clusterings, namely, one that has much smaller optimum. In addition for the min-max variant, the overlapping version admits a simple approximation algorithm, while our algorithm for the non-overlapping version is complex and yields a worse approximation ratio due to the presence of the additional constraint. Somewhat surprisingly, for the problem of minimizing the *sum* of conductances, we found out that allowing overlap does not help. We show how to apply a general technique to *transform* any overlapping clustering into a non-overlapping one with only a modest increase in the sum of conductances. This *uncrossing* technique is of independent interest and may find further applications in the future.

We consider this work as a step toward rigorous comparison of overlapping and non-overlapping clusterings and hope that it stimulates further research in this area.

*IBM T.J.Watson research center. email: rkhandekar@gmail.com.

†Department of Computer Science, Rutgers University-Camden. Partially supported by NSF Award Grant number 434923. email: guyk@crab.rutgers.edu.

‡Google Research, New York, USA. email: mirrokni@gmail.com

1 Introduction

Graph clustering has several fundamental applications including *analyzing social networks* and *efficient distributed computing*:

– As online social networks are popular with hundreds of millions of users, they are becoming a rich source of user-specific data. An important problem on such social networks is the discovery of *communities*. Modeling, discovering and analyzing such communities can help understanding structural properties of these networks and help in potential applications like recommendation systems and advertising.

– Another application of graph clustering is to aid efficient computations on large graphs which model interactions between different elements in a system, e.g., a biological system. Efficient computations on such graphs entail a careful partitioning of the vertices into clusters such that no single cluster is too large, and the number of edges crossing the clusters is small. Each cluster then is stored on a separate machine, and the interaction is carried out via communication between different machines.

Non-overlapping clustering. In this variant, we have a constraint in the clustering problems that the set of clusters should be disjoint. This constraint has been considered in most of the well-studied clustering problems in graph theory and combinatorial optimization literature [18, 7, 25, 2, 31].

Overlapping clustering: While disjoint clusters is a reasonable constraint in some settings, it may not be necessary/appropriate in others as we discuss below. In some settings, like discovering communities in social networks [29, 22], the clusters are naturally overlapping and by restricting our attention to non-overlapping clustering, we may lose valuable information about the structure of communities in a social network [29]. For example, consider a graph with a small number of popular nodes that are well-connected to many other nodes in the graph. These nodes may naturally belong to more than one cluster. For more applications that show that overlapping clustering is more suitable than non-overlapping clustering, see clustering for social networks [29, 22], clustering for distributed computing [28, 3], clustering for inherent multi-assignment clustering [36] and clustering large networks for distributed PageRank computation and performing distributed random walks [3]. For a survey on such models of graph clustering, see the article by Brandes et al. [11] and the references therein. Other than being more appropriate in some clustering scenarios, non-overlapping clustering is often harder to perform in a large scale, as the disjointness constraint is a statement about all of the clusters and requires careful distributed or global implementations". (Note that it is possible to have distributed non-overlapping clustering algorithms, see for example [33]).

As allowing overlap may improve the quality of clusters and has the above advantages in practice, it is natural to assume that overlap will significantly improve and simplify approximation algorithms for minimizing conductance. Now two properties may hold if overlapping clusters are allowed. First, the optimum conductance value may drastically decrease. Second, the approximation algorithm for the overlapping case may become simpler and have a better ratio than if the overlap is not allowed.

In this paper, we partially prove the above intuition to be correct. To this end, we study two natural ways in which one can aggregate the conductances of all the clusters in a clustering – minimizing the *maximum* or minimizing the *sum*. For clustering to minimize the maximum conductance of any cluster, two significant advantages are derived by allowing overlap. The optimum for the overlapping case may be much smaller than the one for non-overlapping clustering. In addition, the approximation algorithm for the overlapping case is simple and has ratio $O(\log n)$. On the other hand the problem of non-overlapping clustering is complex, and thus, the algorithm given for it is also much more involved. Also, this algorithm has a worse approximation factor as can be seen later. On the other hand, for the measure of minimizing the sum of conductances over all clusters, we show a *general uncrossing technique* to transfer an overlapping clustering to a non-overlapping one with small penalty. This implies that the two models are equivalent up to a constant factor with respect to approximation in the sum version. Hence, overlapping does not help much in this case contradicting our initial intuition. We believe that the transformation of overlapping clusters to non-overlapping ones is of independent interest and may find other applications.

We are not aware of a *theoretically rigorous* study that exhibits that allowing overlaps gives improved results compared to non-overlapping clustering. Initializing this study here, we hope that this important topic will be studied from theoretical and practical points of view in the future. Note that our main contributions are theoretical, and our polynomial-time algorithms may not be very practical for very large graphs very, mainly because of the complex dynamic programs involved in them. However we note that different branches of the dynamic program can be implemented in parallel, and thus the dynamic programs can be implemented more efficiently in a distributed fashion.

Follow-up Work. Following the conference version of this paper, there have been various attempts to study the overlapping clustering problem from a theoretical perspective, and present a rigorous study of computing overlapping communities in social networks. For example, Arora et al [4] study the problem of overlapping communities, present an axiomatic approach for defining overlapping social communities, and design approximation algorithms for discovering such communities in a set of randomly generated graphs. Moreover, Balcan et al [8] considered the problem of enumerating and identifying communities generalizing and refining the definition of communi-

ties by Mishra et al[29]. These papers do not explicitly attempt to optimize the cut conductance, but they consider other related notions for quality of clusters like the cut value and density of clustersl [4, 8]. All the recent research about the topic of overlapping clustering imply the need for exploring this topic from a more algorithmic and theoretical perspectivel [4, 8].

1.1 Problem Formulation

Consider an undirected graph $G = (V, E)$ with non-negative edge-weights $w_e \geq 0$. For simplicity, we assume that the weights w_e are integers that are polynomially bounded in $n = |V|$. Our results, however, can be generalized for arbitrary weights as well – we omit the details from this version. For a subset of vertices $S \subset V$, let $\delta(S)$ denote the set of edges in E with exactly one end-point in S. For a vertex $v \in V$, let $\deg(v) = \sum_{e \in \delta(v)} w_e$ denote the total edge-weight incident to v. For a subset $S \subseteq V$, let $\mathrm{vol}(S) = \sum_{v \in S} \deg(v)$.

The *conductance* of a cut $(S, \overline{S} = V \setminus S)$ is defined[1] as $\phi(S) = \frac{\sum_{e \in \delta(S)} w_e}{\min\{\mathrm{vol}(S), \mathrm{vol}(\overline{S})\}}$. We consider several variants of the minimum conductance clustering problem. The input to this problem also includes a volume-budget $B \geq 0$. We assume that the volume-budget is very small compared to the total volume, say $B < \mathrm{vol}(V)/12$ to be concrete. Putting a bound on the volume makes sense for a couple of reasons: One reason is that if we want to cluster the graph for the purpose of performing some distributed computation, it is important that each cluster fits on one machine, and amount of memory to store the cluster is directly proportional to the volume of the cluster. This is captured by putting an upper bound on the volume of each cluster. Another reason to put a bound on the volume of the clusters is that it is harder to understand and conceptualize huge clusters, and to make sure that the clustering algorithm generates clusters that are easily understood and interpretable, we put an upper bound on the size or volume of each cluster.

The output is a *clustering* of the vertices, i.e., subsets $S_1, \ldots, S_k \subset V$ such that $V = \cup_i S_i$ and $\mathrm{vol}(S_i) \leq B$ for all i. Following are the variants of the basic clustering problem we consider. (I) In an Overlap clustering, the subsets S_i are allowed to intersect with each other, while in a Nonoverlap clustering, the subsets S_i must be disjoint, thus forming a vertex partitioning. (II) In a Bound (or bounded-count) clustering, we require that the total number of clusters k is at most given count-budget k, while in an Unbound (or unbounded-count) clustering, there is no bound on the number of clusters formed. (III) In a Sum (or min-Sum) clustering, the objective is to minimize the total conductance of the clusters: $\min \sum_{1 \leq i \leq k} \phi(S_i)$, while in a Max (or min-max) clustering, the objective is to minimize the maximum conductance of a cluster: $\min \max_{1 \leq i \leq k} \phi(S_i)$.

[1] Alternatively, we can define the conductance (or, more appropriately the *sparsity*) of a cluster as $\sum_{e \in \delta(S)} w_e / \min\{|S|, |\overline{S}|\}$. Our results also hold for this definition.

1.2 Related work

Partitioning a graph into two parts Most approximation algorithms for partitioning problems require the vertices to be partitioned into two parts only. Some examples follow. In [19], a graph $G(V, U)$ and an integer k is given. The goal is to find a set $U \subseteq V$ of size k so that $e(U)$, namely, the number of edges with both endpoints in U, is maximized.

Another example is the *Sparsest cut* problem in which the goal is to split the graph into two disjoint parts (thus defining a cut) and maximize the number of separated pairs over the cost of the cut (see [7]). The unweighted version admits an $O(\sqrt{\log n})$ approximation [7]. The weighted case admits a slightly worse approximation. See [5] and references therein.

The *Bisection* problem is to partition V into two equal and disjoint parts A and B, and minimize the number of edges between A and B. The problem admits an $O(\log n)$ approximation [32].

Partitioning problems into more than two parts: In the *Minimum cut*, problem given an edge-weighted graph and a collection of pairs $\{(s_i, t_i)\}_{i=1}^{k}$, the goal is to find a subset $E' \subseteq E$ of minimum cost so that in $V \setminus E'$ there is no s_i-t_i path for any i. Clearly $G \setminus E'$ may have many parts. Garg et. al. [21] showed that the Minimum multicut problem is at least as hard to approximate as the well-known vertex-cover problem even if the underlying graph is a star. This implies that unless $P = NP$, it is hard to approximate the undirected multicut problem on stars within a factor better than $10\sqrt{5} - 21$ [17]. Garg et. al. [21] also gave a 2 approximation for Minimum multicut in trees via the primal dual approach. The best known approximation for Undirected multicut in general undirected graphs is $O(\log n)$ [20].

In the *Directed multicut* problem the goal is to discard min cost set of edges E' so that for every s_i, t_i there is no *directed* s_i, t_i path $G'(V, E - E')$. In [16] it is shown that unless $NP \subseteq ZPP$ the Directed multicut problem admits no $2^{\log^{1-\epsilon} n}$ ratio for any constant ϵ. The first non-trivial approximation for Directed multicut was an $O(\sqrt{n \log n})$ ratio by Cheriyan et. al. [14]. This ratio was improved to an $O(\sqrt{n})$ by Gupta [23]. Recently, the $O(\sqrt{n})$ barrier was broken by Agrawal et. al. [1] to an $\tilde{O}(n^{11/23})$ ratio approximation[2]. In [26], an $O(n^{2/3}/opt^{1/3})$ approximation algorithm for uniform costs directed multicut, is presented. Here opt is the optimum value. If the optimum is at least $n^{0.566}$, the ratio of [26] is better than the one of [1]. The [26] algorithm is also the only non-trivial *combinatorial* approximation algorithm for Directed multicut. For more papers on cut problems see [27, 6, 15, 13, 35, 10]. Some of these papers were an inspiration for this paper.

No previous papers gave approximation algorithm for conductance. The only slightly related result is by Bansal et. al. [9]. This problem considered is, given a graph G and a number k,

[2]The notation $\widetilde{O}(f(n))$ ignores the factors polylogarithmic in n.

min-sum	overlap	no-overlap
bounded-count	SumOverlapBound $O(\log n)$ (with $O(K)$ clusters)	SumNonoverlapBound $O(\log n)$ (with $O(K)$ clusters)
unbounded-count	SumOverlapUnbound $O(\log n)$	SumNonoverlapUnbound $O(\log n)$
min-max	**overlap**	**no-overlap**
bounded-count	MaxOverlapBound $O(\log n)$ (with $O(K \log n)$ clusters)	MaxNonoverlapBound $O(\log^4 n \log\log n)$ (with $O(K)$ clusters)
unbounded-count	MaxOverlapUnbound $O(\log n)$	MaxNonoverlapUnbound $O(\log^4 n \log\log n)$

Table 1: The approximation ratios obtained by polynomial-time algorithms for different variants of the problem. The volume-budget B (resp. the count-budget K, if given) is violated by at most a constant factor (resp. given factor). The most complex technical contribution is shaded.

divide the vertices into k parts and minimize the maximum number of edges leaving any set in the partition. The number of edges leaving a set in the partition is related to conductance, but the problems are very different, still. For two more problems that admit non trivial approximation and require to partition the graph into more than two parts see [12, 34]. Generally speaking, it seems that approximating partitioning problems into many parts is a quite complex subject.

1.3 Our results

Considering all the variants, we define eight minimum conductance clustering problems. The notation and results for these problems is summarized in Table 1. While we have general technical contributions in comparing overlapping vs. non-overlapping clustering discussed above, and in the uncrossing technique, the algorithm for minimizing the maximum conductance in the non-overlapping case, is the most complex algorithm in the paper. To solve this problem, we begin by a tree decomposition of the graph — but now, we need a single tree of Räcke [30] or Harrelson et al. [24] that preserves all the cuts *simultaneously*. This is required since the non-overlapping problems, unlike overlapping or min-sum problems, is inherently *global* in nature. [3] The difficulty in designing an approximation algorithm for MaxNonoverlapBound is not only in the dynamic programming, but also in proving some non-trivial structural property of a near-optimal solution

[3] A similar consideration does not hold for the non-overlapping version of the min-sum problem since the overlapping and non-overlapping versions turn out to be equivalent for the min-sum problem. This is explained in detail in Section 2.

in terms of how many subtrees of different "(volume, cut-value) types" can a near-optimum clustering contain. Our dynamic program then forms near-optimal clusters by taking the union of the "correct" number of subtrees of certain (volume, cut-value) types (Section 3).

The hardest problem to approximate seems to be MaxNonoverlapBound. All the other variants of the problem admit an $O(\log n)$-approximation algorithm. These results are summarized in Table 1 and presented in Section 4. These results are mainly based on the tree decomposition of Räcke [31] that embeds the given graph into (a distribution on) trees and preserves all the cuts within a logarithmic factor. We then present constant-factor approximations to these problems on trees based mainly using a dynamic program over trees.

Remark 1 *It is easy to see why a distribution on tree-embeddings that preserves any cut within a logarithmic factor in expectation is enough to get logarithmic approximations for the min-sum versions of the problem. Somewhat surprisingly, it is also enough to get logarithmic approximation for min-max and* overlapping *version of the problem. The reason, roughly speaking, is that we compute separate (and possibly overlapping) clusters covering different parts of the graph; and for this, we do not necessarily need a* single *tree that preserves all the cuts in the graph* simultaneously. *This was clearly not the case for min-max non-overlapping version.*

Since we claim that in the min-max conductance variant with overlapping, our algorithm is much simpler than the non-overlap algorithm, it seems appropriate to shortly describe the overlapping algorithm. In order to get a logarithmic approximation for minimum MaxOverlapBound, we design a set-cover-type greedy algorithm as follows: By guessing the value of OPT, we iteratively find a set $S_t \subseteq V(G)$ with the conductance of at most OPT which maximizes the total weight of *uncovered nodes*, and add this set S_t to the output. In order to identify *cost-effective* clusters as a step of the above set-cover-type algorithm, we solve a natural dynamic program over trees.

2 Comparing the optima of overlapping and non-overlapping clusterings

In this section, we show that *min-max* conductance overlapping and non-overlapping optimization problem might be quite different, but *min-sum* conductance overlapping and non-overlapping problems are similar in terms of their approximability. We start by showing a large gap between the optimal solutions of the *min-max* conductance overlapping clustering problem versus the *min-max* conductance non-overlapping clustering problem. If you do not find this example particularly natural, let us state that it is our experience that most times, if you have any example for a gap, this implies that there is a "natural" example for the gap. The next example may not correspond

to a natural social network. However, it is our experience that once we establish the gap between the two variants its highly likely that a more representative example with the same gap exists. For completeless, we also provide an more natural family of examples showing a smaller gap for min-max conductance between the overlapping and non-overlapping clustering.

Lemma 2.1 *There exists an infinite family of graphs $G = (V, E)$ such that there exists an instance of min-max conductance clustering problem on G such that*

1. *the optimum value of the* overlapping *version is $O(|V|^{-2/3})$,*
2. *the optimum value of the* non-overlapping *version is $\Omega(1)$, even if we violate the budget by an $\Omega(|V|^{1/3})$ factor.*

Proof: For an integer k, let $G = (V, E)$ be the disjoint union of a clique K_k on k vertices and a 3-regular expander H on k^3 vertices. Let the weight of each edge in G be 1. Denote $n = |V| = k^3+k$ and let the budget on the volume of clusters be $B = k(k-1)+3$.

We first prove item 1. For each vertex $v \in H$, define a cluster C_v to be the union of K_k and v. Note that $\mathrm{vol}(C_v) = k(k-1)+3$ and $\phi(C_v) = 3/B = \Theta(1/k^2) = \Theta(n^{-2/3})$. These clusters are over-lapping and satisfy property in item 1. Now for item 2, we prove that as long as the budget is not violated by $\frac{3k^3}{2B}$ factor, the optimum value of the non-overlapping clustering is $\Omega(1)$. Consider the optimum clustering. First note that the budget violation considerations imply that any cluster contains at most $k^3/2$ vertices from H. Now since the number of vertices in H is k^2 times the number of vertices in K_k, there exists a cluster C such that the number of vertices in $C \cap H$ is $p \geq 1$ and the number of vertices in $C \cap K_k$ is at most $\lfloor p/k^2 \rfloor$. Since H is an expander, the cut capacity of this cluster is $\Omega(p)$. On the other hand, the volume of C is at most $3p + p(k-1)/k^2$. Thus the conductance of C is $\Omega(1)$ as desired. ∎

Despite its intuitive appeal, overlap does not always help. The following lemma shows that the *min-sum* conductance overlapping and non-overlapping clustering problems are essentially identical, up to a constant in the approximation factor.

Lemma 2.2 *Any solution for the* min-sum *conductance* overlapping *problem, of objective value ϕ and maximum cluster volume b, can be converted into a solution for the* min-sum *conductance* non-overlapping *problem of objective value at most 2ϕ and maximum cluster volume at most $3b/2$.*

Proof: Let $S_1, \ldots, S_k$ be a solution for the overlapping problem with $\phi = \sum_i \phi(S_i) \geq \sum_i w(\delta(S_i))/b$. Now we systematically "uncross" the sets $\{S_i\}$ as follows. For any two intersecting sets X and Y, we first observe that $w(\delta(X))+w(\delta(Y)) \geq \min\{w(\delta(X))+w(\delta(Y \setminus X)), w(\delta(X \setminus Y))+w(\delta(Y))\}$. Thus we can replace X and Y with either X and $Y \setminus X$, or $X \setminus Y$ and Y without increasing

the total cut value. We can in polynomial time, uncross all the sets $\{S_i\}$ to get a family $\{T_j\}$ of non-overlapping clusters. Note that $\sum_i w(\delta(S_i)) \geq \sum_j w(\delta(T_j))$ and $\mathrm{vol}(T_j) \leq b$ for all j. Now we successively merge the clusters $\{T_j\}$ as follows. For any two clusters X and Y with $\mathrm{vol}(X), \mathrm{vol}(Y) \leq b/2$, replace X and Y with $X \cup Y$. Thus finally, we are left with all clusters of volume between $b/2$ and b and at most one small cluster of volume less than $b/2$. We can merge this small cluster with any of the other clusters. This way, we form non-overlapping clusters $\{U_1, \ldots, U_p\}$ with $\sum_j w(\delta(T_j)) \geq \sum_l w(\delta(U_l))$ and $b/2 \leq \mathrm{vol}(U_l) \leq 3b/2$ for all l. Note that $\{U_l\}$ is $\sum_l \phi(U_l) \leq 2\sum_l w(\delta(U_l))/b \leq 2\sum_j w(\delta(T_j))/b \leq 2\sum_i w(\delta(S_i))/b \leq 2\phi$, as desired. ■

In the above proof, the clusters used in the gap example are not connected. The following is an example showing a smaller gap with more natural clusters. Consider a graph with $k^{1.5}$ cliques of size k, a random (Erdos-Renyi) graph of k^3 vertices with expected degree $k^{0.5}$. Each clique connects to a (disjoint) set of size $k^{1.5}$ in the random graph with $k^{1.5}$ edges. The volume constraint is still k^2. For this example, it is not hard to see that for non-overlapping, one still gets only constant conductance. However for overlapping it's easy to get $k^{-0.5}$, i.e.,for each cluster, take k vertices in the random graph and a clique. For this instance the gap is smaller but at least the graph is connected.

As a consequence, a ρ-approximation algorithm for one problem can be used to obtain an $O(\rho)$-approximation algorithm for the other problem, provided we are willing to violate the budgets by a constant factor. We appreciate however that although a factor of two may not matter in "theory", it may have an impact in practice.

3 The non-overlapping *min-max* conductance clustering

In this section, we focus on our most involved result, the non-overlapping *min-max* conductance clustering MaxNonoverlapUnbound and present a poly-log approximation ratio for this problem. We first note that the two variants (bounded-count and unbounded-count) of this problem are essentially the same, provided we violate the volume-budget B and the count-budget K by a constant factor. First note that $B \cdot K \geq \mathrm{vol}(V)$ must hold for a feasible solution to exist. Now any solution for the unbounded-count variant can be transformed into a solution for the bounded-count without increasing the min-max objective as follows. Consider a clustering $S_1, \ldots, S_k$ for the unbounded-count variant where $b = \max_i \mathrm{vol}(S_i) = O(B)$. Now observe that the conductance of the union two clusters is at most the maximum conductance of the two clusters: $\phi(S_i \cup S_j) \leq \max\{\phi(S_i), \phi(S_j)\}$. Thus we can take union of the clusters till all clusters have volume between $B/2$ and $2 \cdot \max\{b, B\}$. This process results in $O(K)$ clusters with maximum conductance at most that of the original clustering.

Theorem 3.1 *There is a polynomial-time $O(\log^4 n \log\log n)$-approximation for the no-overlap min-max conductance clustering, in which the budget is violated by a at most constant factor.*

Outline of our approach. We reduce the problem on the general graph to trees by using the techniques of Räcke [31] and Harrelson et al. [24] thereby losing a factor of $O(\log^2 n \log\log n)$ in the approximation. The leaves in the tree are in one-to-one correspondence with the vertices in the original graph. With each leaf v in the tree, we associate a weight $w_v = \deg(v)$, the weighted degree of v in the original graph. The problem on trees is to partition the leaves into disjoint clusters such that each cluster has weight at most B and the maximum conductance of any cluster S, defined as the ratio of the min-cut, in the tree, separating S from rest of the leaves to the weight of S, is minimized. We compute a logarithmic approximation for this problem on trees by using a complex dynamic program. Before describing the dynamic program, we study the structure of the optimum solution. Note that the edge-weights w_e are integers between 1 and B where B is polynomially bounded in n. Since we violate the budget B by a constant factor anyway, we assume that $B = 2^b$ for some integer $b = O(\log n)$. Consider the optimum clustering on the tree $S_1^*, \ldots, S_l^*$. If we remove the edges in the min-cuts separating each S_i^* from rest of the leaves, the tree gets decomposed into disjoint subtrees. Let $T_1^*, \ldots, T_t^*$ be the disjoint subtrees thus formed. Let $\mathrm{vol}(T_i^*)$ denote the total volume of the leaves in tree T_i^* and let $\mathrm{cut}(T_i^*)$ denote the capacity of the cut separating T_i^* from rest of the tree. Also let $\mathrm{ratio}(T_i^*) = \mathrm{cut}(T_i^*)/\mathrm{vol}(T_i^*)$. We say that a tree T_i^* belongs to class k if $2^k \cdot \textsc{opt} \leq \mathrm{ratio}(T_i^*) < 2^{k+1} \cdot \textsc{opt}$ where OPT is the optimum value of the maximum conductance for the tree instance. In the algorithm, we first guess the value OPT. We then estimate the total volume of trees T_i^* in the optimum solution that belong to each class. Our dynamic program then decomposes the tree into the correct volume of trees from different classes and greedily combines them to form a clustering. This clustering naturally induces a clustering in the original graph.

Details of the approach. We now describe our approach in more details. We first state the result of Harrelson, Hildrum, and Rao [24] more formally. A tree decomposition $\mathcal{T}$ of a graph $G = (V, E)$ is described by a series of hierarchical partitions of the vertex set V of G. The nodes of $\mathcal{T}$ correspond to the subsets of V. Consider a series of partitions $\Pi_0, \ldots, \Pi_d$ where partition Π_{i+1} is a refinement of partition Π_i. The partition Π_0 corresponds to a single set V while the partition Π_d corresponds to the set of singletons $\{v\}$ where $v \in V$. These partitions give rise to a tree $\mathcal{T}$ naturally. The root node of $\mathcal{T}$ is V itself. The nodes in layer i are the sets in Π_i and the leaves correspond to the sets in Π_d, i.e., the vertices in V. The edges of the tree go between the consecutive layers and are given by set inclusion. The weights of the edges of the tree are given as follows. For a setpair (S, T), where $S \subset T$, $S \in \Pi_{i+1}$, and $T \in \Pi_i$, the weight $w(S, T) = w_G(\delta(S))$ is defined to be the weight of the cut $(S, \overline{S})$ in the graph G. For $S \subset V$, define $w_{\mathcal{T}}(S, \overline{S})$ to be the minimum cut in $\mathcal{T}$ that separates leaves in S from the leaves in $\overline{S}$. Harrelson et al. [24] proved the following theorem.

Theorem 3.2 (Harrelson et al. [24]) *In time polynomial in $n = |V|$, one can compute a tree decomposition $\mathcal{T}$ with depth $d = O(\log n)$ such that for any $S \subset V$, we have*

$$w_G(\delta(S)) \leq w_{\mathcal{T}}(S, \overline{S}) \leq O(\log^2 n \log\log n) \cdot w_G(\delta(S)).$$

Remark 2 *Harrelson et al. [24] did not state their theorem as above. They showed how to compute a decomposition $\mathcal{T}$ such that any multicommodity-flow, that can be routed in G with (edge) congestion 1, can be routed in $\mathcal{T}$ with congestion at most 1 and any multicommodity-flow (between leaves of $\mathcal{T}$), that can be routed in $\mathcal{T}$ with congestion 1, can be routed in G with congestion $O(\log^2 n \log\log n)$. It is easy to see that this implies Theorem 3.2. Fix a subset $S \subset V$. The multicommodity-flow given by demands $r_{ij} = w_e$ for $e = (i,j) \in \delta(S)$ can be routed in G with congestion 1. Thus it can be routed in $\mathcal{T}$ with congestion at most 1. Hence $w_G(\delta(S)) \leq w_{\mathcal{T}}(S, \overline{S})$. On the other hand, consider the maximum flow (of total value $w_{\mathcal{T}}(S, \overline{S})$) that can be routed between leaves S and leaves $\overline{S}$ in $\mathcal{T}$. Fix a flow-path decomposition of this flow. This decomposition gives multicommodity-flow demands between S and $\overline{S}$ that can be routed in $\mathcal{T}$ with congestion 1. Thus it can be routed in G with congestion $O(\log^2 n \log\log n)$. Since all of this flow must cross the cut $\delta(S)$ in G, we get that $w_G(\delta(S)) \geq w_{\mathcal{T}}(S, \overline{S})/O(\log^2 n \log\log n)$, implying the theorem. The same connection holds also for the result of Räcke [31] (see Theorem 4.1).*

Let $S_1^*, \ldots, S_l^*$ be the optimum clustering in G and let $\text{OPT} = \max_i \phi(S_i^*)$ be the value of the optimum. Consider the minimum cuts in $\mathcal{T}$ that separate the leaves in S_i^* from the leaves in $\overline{S}_i^*$ for each i. If we remove the tree-edges that appear in any of these min cuts from $\mathcal{T}$, the tree $\mathcal{T}$ gets decomposed into a collection of disjoint subtrees $T_1^*, \ldots, T_t^*$. For each subtree T, let $l(T)$ denote the set of leaves of $\mathcal{T}$ present in T, $\text{vol}(T) = \text{vol}(l(T))$ be the total volume of T, and $\text{cut}(T)$ denote the total weight of the tree-edges needed to separate T from rest of the tree $\mathcal{T}$. Here we consider only those subtrees T such that $\text{vol}(T) > 0$. Define $\text{ratio}(T) = \text{cut}(T)/\text{vol}(T)$ to be the ratio of T. We now define a notion of a class of a tree based on its ratio as follows.

Definition 3.1 *We say that a tree T belongs to class 0, denoted by $\mathcal{C}_0$, if* $\text{ratio}(T) < 2 \cdot \text{OPT}$. *For $k \geq 1$, we say that a tree T belongs to class k, denoted by $\mathcal{C}_k$, if* $2^k \cdot \text{OPT} \leq \text{ratio}(T) < 2^{k+1} \cdot \text{OPT}$.

Thus there are at $O(\log \text{vol}(V)) = O(\log n)$ classes. We now prove an important lemma about the trees T_i^* obtained from the optimum solution.

Lemma 3.3 *We have (i) The volume of any tree in class $\mathcal{C}_k$ is at most $2^{-k} \cdot B$, or formally, $\text{vol}(T_i^*) \leq 2^{-k} \cdot B$ for any tree $T_i^* \in \mathcal{C}_k$ for any $k \geq 0$, and (ii) The volume of all trees in class $\mathcal{C}_k$ or higher is at most $2^{-k} \cdot \text{vol}(\mathcal{T})$, or formally, $\sum_{l \geq k} \sum_{i: T_i^* \in \mathcal{C}_l} \text{vol}(T_i^*) \leq 2^{-k} \cdot \text{vol}(\mathcal{T})$ for any $k \geq 0$.*

Proof: We first prove item 1. Note that item 1 trivially holds for $k = 0$ since any tree is contained in a single cluster with volume at most B. For $k > 0$, if the inequality in item 1 does not hold, then

$\mathrm{cut}(T_i^*) = \mathrm{vol}(T_i^*) \cdot \mathrm{ratio}(T_i^*) > 2^{-k} \cdot B \cdot 2^k \cdot \textsc{opt} = B \cdot \textsc{opt}$. This implies that the total cut of the optimum cluster containing T_i^* is more than $B \cdot \textsc{opt}$. Since the volume of this cluster is at most B, its conductance is more than OPT, which is a contradiction. We now prove item 2. Note that item 2 trivially holds for $k = 0$. For $k > 0$, if the inequality in item 2 does not hold, then

$$\sum_{l \geq k} \sum_{i: T_i^* \in \mathcal{C}_l} \mathrm{cut}(T_i^*) > 2^{-k} \cdot \mathrm{vol}(\mathcal{T}) \cdot 2^k \cdot \textsc{opt} = \mathrm{vol}(\mathcal{T}) \cdot \textsc{opt}.$$

This is a contradiction since it implies that $\sum_i \mathrm{cut}(T_i^*) / \sum_i \mathrm{vol}(T_i^*) > \textsc{opt}$, which in turn means that the conductance of at least one of the optimum clusters is more than OPT. ■

The above lemma motivates the following definition.

Definition 3.2 *A partition of $\mathcal{T}$ into disjoint subtrees $T_1, \ldots, T_l$ is called* admissible *if (i) $\mathrm{vol}(T_i) \leq 2^{-k} \cdot B$ for any tree $T_i \in \mathcal{C}_k$ for any $k \geq 0$, and (ii) $\sum_{i: T_i \in \mathcal{C}_k} \mathrm{vol}(T_i) \leq 2^{-k} \cdot \mathrm{vol}(\mathcal{T})$ for any $k \geq 0$.*

The following lemma presents an important property of an admissible partition.

Lemma 3.4 *Given any admissible partition $T_1, \ldots, T_l$ of $\mathcal{T}$, the sets of leaves $l(T_i)$ of these trees can be combined, in polynomial time, to form clusters of vertices in G, each of volume at most $10B$ and such that the maximum conductance of any cluster is $O(\textsc{opt} \cdot \log n)$.*

Proof: Item 1 is proved as in Lemma 3.3 We now prove item 2. Note that item 2 trivially holds for $k = 0$. For $k > 0$, if the inequality in item 2 does not hold, then

$$\sum_{l \geq k} \sum_{i: T_i^* \in \mathcal{C}_l} \mathrm{cut}(T_i^*) > 2^{-k} \cdot \mathrm{vol}(\mathcal{T}) \cdot 2^k \cdot \textsc{opt} = \mathrm{vol}(\mathcal{T}) \cdot \textsc{opt}.$$

This is a contradiction since it implies that $\sum_i \mathrm{cut}(T_i^*) / \sum_i \mathrm{vol}(T_i^*) > \textsc{opt}$, which in turn means that the conductance of at least one of the optimum clusters is more than OPT. We combine the sets of leaves $l(T_i)$ of the trees T_i into $K = \lceil \mathrm{vol}(V)/4B \rceil$ clusters as follows. For each class $\mathcal{C}_k$, we distribute the trees $T_i \in \mathcal{C}_k$ into K clusters as follows. Consider the trees in $\mathcal{C}_k$ in arbitrary order and assign the next tree in $\mathcal{C}_k$ to a cluster that has the minimum volume of the trees in class $\mathcal{C}_k$. Below we show that this clustering satisfies the conditions claimed. Now consider any of these K clusters. The volume of trees in a class $\mathcal{C}_k$ in this cluster is at most $2^{-k} \cdot \mathrm{vol}(V)/(\mathrm{vol}(V)/4B) + 2^{-k} \cdot B \leq 5B \cdot 2^{-k}$. This holds due to the way trees in class $\mathcal{C}_k$ are assigned to clusters. Thus the total volume of this cluster is at most $5B \sum_k 2^{-k} < 10B$. We now argue that the conductance of each of these K clusters is $O(\textsc{opt} \cdot \log n)$. First of all, we argue that the volume of any cluster is at least B. Let V_k denote the total volume of trees in class $\mathcal{C}_k$, thus $\sum_k V_k = \mathrm{vol}(V)$. Again from the assignment of trees from different classes to clusters, we get that the total volume of any cluster is at least

$$\sum_k \left(\frac{V_k}{\lceil \mathrm{vol}(V)/4B \rceil} - 2^{-k} \cdot B \right) \geq \sum_k \left(\frac{V_k}{\mathrm{vol}(V)/3B} - 2^{-k} \cdot B \right) \geq 3B - 2B = B.$$

The first inequality holds if $\lceil \mathrm{vol}(V)/4B \rceil \leq \mathrm{vol}(V)/3B$ which, in turn, holds if $12B \leq \mathrm{vol}(V)$.

Next note that the total cut value of any cluster $\mathcal{C}$ is at most

$$\begin{aligned}
\sum_k \sum_{T \in \mathcal{C} \cap \mathcal{C}_k} \mathrm{cut}(T) &= \sum_k \sum_{T \in \mathcal{C} \cap \mathcal{C}_k} \mathrm{ratio}(T) \cdot \mathrm{vol}(T) \\
&< \sum_k 2^{k+1} \cdot \textsc{opt} \sum_{T \in \mathcal{C} \cap \mathcal{C}_k} \mathrm{vol}(T) \\
&\leq \sum_k 2^{k+1} \cdot \textsc{opt} \cdot 5B \cdot 2^{-k} \\
&\leq \sum_k 10\textsc{opt} \cdot B \\
&= O(\textsc{opt} \cdot B \cdot \log n).
\end{aligned}$$

The last inequality follows since the total number of classes is $O(\log n)$. This completes the proof. ■

The above lemma, in particular, implies that computing an admissible partition of $\mathcal{T}$ yields $O(\log^3 n \log\log n)$ approximation to our original problem on the graph. Here an $O(\log^2 n \log\log n)$ factor comes from Theorem 3.2 and another $O(\log n)$ factor comes from the above lemma.

The dynamic program. We now present a dynamic program based algorithm for finding an admissible partition of $\mathcal{T}$. To convey the basic intuition behind our approach, we first describe a dynamic program that runs in $n^{O(\log n)}$ time. Later we sketch how to turn it into a polynomial-time algorithm, losing another factor of $O(\log n)$ in the approximation ratio.

In the dynamic program, we first compute and store some *tables* for the leaves of $\mathcal{T}$. We then show how to use the tables for the children of a node to compute the table for the node itself. Finally, the table for the root node is used to compute an admissible partition of $\mathcal{T}$. Let $C = O(\log n)$ denote the total number of classes.

Fix a node u of the tree $\mathcal{T}$. The table for u has the following form. Consider a vector $\vec{v} = [c, v, v_0, v_1, \ldots, v_C]$ where $c, v_0, \ldots, v_C$ are integers in the range $[0, \mathrm{vol}(V)]$. Note that there are at most $\mathrm{vol}(V)^{O(\log \mathrm{vol}(V))} = n^{O(\log n)}$ such vectors.

Definition 3.3 *The vector $\vec{v}$ is called* valid *for node u if the subtree hanging below u can be partitioned into disjoint subtrees $T_0, T_1, \ldots, T_l$ such that (i) The node u belongs to the tree T_0 and $c = \mathrm{cut}(T_0), v = \mathrm{vol}(T_0)$, and (ii) For each class $0 \leq k \leq C$, the total volume of trees $T_1, \ldots, T_l$ that belong to class $\mathcal{C}_k$ is exactly v_k, i.e., $\sum_{i:T_i \in \mathcal{C}_k} \mathrm{vol}(T_i) = v_k$ for all $k \geq 0$.*

The table of u stores a list of all valid vectors $\vec{v}$ and the corresponding partition $T_0, \ldots, T_l$ as a certificate that makes $\vec{v}$ valid. We first note that it is easy to compute the table for the leaves of

$\mathcal{T}$. We now explain how to compute the table for a node v given the tables of all of its children $u_1, \ldots, u_p$. If $p = 1$, it is easy to compute the table of v from that of u_1. Consider the case $p = 2$ now. For each vector $\vec{v} = [c, v, v_0, v_1, \ldots, v_C]$, in order to determine if there exists a partition $T_0, \ldots, T_l$ that makes $\vec{v}$ valid, we consider several cases. These cases are based on whether T_0 would have components from both the subtrees hanging from u_1 and u_2 (case 1) or just one (case 2). For case 1, we consider all possible ways of decomposing it into two vectors $\vec{v}^1 = [c^1, v^1, v_0^1, v_1^1, \ldots, v_C^1]$ and $\vec{v}^2 = [c^2, v^2, v_0^2, v_1^2, \ldots, v_C^2]$ such that $\vec{v} = \vec{v}^1 + \vec{v}^2$ where the addition is done component-wise. Note that there are $\mathrm{vol}(V)^C = n^{O(\log n)}$ such decompositions. We find a decomposition (if one exists) such that $\vec{v}^1$ is valid for u_1 and $\vec{v}^2$ is valid for u_2. If these exists such a decomposition, we mark $\vec{v}$ valid for v and compute the corresponding partition for $\vec{v}$ as follows. We recover the decompositions $T_0^1, \ldots, T_{l_1}^1$ for $\vec{v}^1$ from the table of u_1 and $T_0^2, \ldots, T_{l_2}^2$ for $\vec{v}^2$ from the table of u_2. We define $T_0 = T_0^1 \cup T_0^2 \cup \{(v, u_1), (v, u_2)\}$, i.e., the tree formed by taking the union of T_0^1, T_0^2, and the edges (v, u_1) and (v, u_2). The decomposition for $\vec{v}$ is then $T_0, T_1^1, \ldots, T_{l_1}^1, T_1^2, \ldots, T_{l_2}^2$.

If there is no such decomposition, we try case 2. Assume that the tree T_0 contains a subtree T_0^1 from the subtree hanging from u_1 and the edge (v, u_1); thus the edge (v, u_2) contributes to $\mathrm{cut}(T_0)$. We consider all possible ways of decomposing $\vec{v}$ into two vectors $\vec{v}^1$ and $\vec{v}^2$ such that the following holds: $c = c^1 + w_{(v,u_2)}$, $v = v^1$, $c^2 = 0$, $v^2 = 0$, and $v_k = v_k^1 + v_k^2$ for all classes k. If there exists such a decomposition, we mark $\vec{v}$ valid and store the appropriate tree partition that makes $\vec{v}$ valid. If there is no such decomposition, we try the same by exchanging the roles of u_1 and u_2. If we do not succeed to mark $\vec{v}$ valid in the above cases, we conclude that $\vec{v}$ is not valid.

The case of $p \geq 3$ children is handled similarly. Conceptually, we can make $\mathcal{T}$ binary by adding dummy edges of weight zero and work with the previous cases. This is equivalent to processing the children $u_1, \ldots, u_p$ from left to right one by one. We first determine all the valid vectors for the subtree which is a union of subtrees hanging from u_1 and u_2 using the $p = 2$ case. We then determine all the valid vectors for the subtree which is a union of subtrees hanging from u_1, u_2, and u_3 using the $p = 2$ case and the previously computed information. We repeat this till we consider all the children of v. Thus we can compute the entire table for v.

In the end, to find an admissible partition of $\mathcal{T}$, we determine if there is a vector $\vec{v}$ (with corresponding partition $T_1, \ldots, T_l$) that is valid for the root node and that satisfies the two conditions in the definition of admissible partition: $\mathrm{vol}(T_i) \leq 2^{-k} \cdot B$ for all $T_i \in \mathcal{C}_k$ and $v_k \leq 2^{-k} \cdot \mathrm{vol}(\mathcal{T})$ for all k.

Making the dynamic program run in polynomial time. We can improve the running time from $n^{O(\log n)}$ to polynomial time if we lose a factor of $O(\log n)$ in the approximation guarantee. This is done using the following main observations: (i) The depth of the tree $\mathcal{T}$ is $O(\log n)$,

say $a \log n$ for some constant $a > 0$, (ii) The number of classes can be reduced from $O(\log n)$ to $O(\log n / \log\log n)$ by defining class 0 to be the set of trees T with $\text{ratio}(T) < (\log n) \cdot \text{OPT}$ and class $k \geq 1$ to be set of trees T with $(\log n)^k \cdot \text{OPT} \leq \text{ratio}(T) \leq (\log n)^{k+1} \cdot \text{OPT}$. This is the place where we lose another factor of $O(\log n)$ in the approximation ratio, and (iii) The coordinates in vectors $\vec{v}$, considered in the dynamic program, instead of ranging over *all* integers in $[0, \text{vol}(V)]$, now range over powers of $(1+\delta)$ in $[0, \text{vol}(V)]$ where $\delta = 1/(a \log n)$. Note that there are $O(\log^2 n)$ distinct powers of $(1+\delta)$ in this interval. Thus the number of distinct vectors $\vec{v}$ is $O(\log^2 n)^{O(\log n/\log\log n)} = n^{O(1)}$.

More formally we show that the dynamic programming algorithm described for no-overlap min-max conductance clustering, can be modified to run in polynomial time. As it is now its running time is time $n^{O(\log n)}$ since the size of each table can be as large as this. We can make the program run in polynomial time provided we are willing to lose a factor of $O(\log n)$ in the approximation guarantee as outlined below. This reduction in the running time is based on the following main observations:

- The depth of the tree $\mathcal{T}$ is $O(\log n)$, say $a \log n$ for some constant $a > 0$.
- The number of classes can be reduced from $O(\log n)$ to $O(\log n / \log\log n)$ by defining class 0 to be the set of trees T with $\text{ratio}(T) < (\log n) \cdot \text{OPT}$ and class $k \geq 1$ to be set of trees T with $(\log n)^k \cdot \text{OPT} \leq \text{ratio}(T) \leq (\log n)^{k+1} \cdot \text{OPT}$. This is the place where we lose another factor of $O(\log n)$ in the approximation ratio.
- The coordinates in vectors $\vec{v}$, considered in the dynamic program, instead of ranging over *all* integers in $[0, \text{vol}(V)]$, now range over powers of $(1+\delta)$ in $[0, \text{vol}(V)]$ where $\delta = 1/(a \log n)$. Note that there are $O(\log^2 n)$ distinct powers of $(1+\delta)$ in this interval. Thus the number of distinct vectors $\vec{v}$ is $O(\log^2 n)^{O(\log n/\log\log n)} = n^{O(1)}$.

For every node $u \in \mathcal{T}$, we define its *height* as follows. The height of the root node is $a \log n$ and the height of any node $u \in \mathcal{T}$ is defined as one less than the height of its parent in $\mathcal{T}$. The definition of valid vectors is modified as below.

Definition 3.4 *The vector $\vec{v}$ is called* valid *for node u of height h if the subtree hanging below u can be partitioned into disjoint subtrees $T_0, T_1, \ldots, T_l$ such that*

- *The node u belongs to the tree T_0 and $c \leq \text{cut}(T_0) < c(1+\delta)^h, v \leq \text{vol}(T_0) < v(1+\delta)^h$.*
- *For each class $0 \leq k \leq C$, the total volume of trees $T_1, \ldots, T_l$ that belong to class $\mathcal{C}_k$ is in the range $[v_k, v_k(1+\delta)^h)$.*

The dynamic program proceeds as before. For each node $u \in \mathcal{T}$, we determine all valid vectors $\vec{v}$ and the corresponding certificates that make them valid. We say that a number v is known within a precision of $(1+\delta)^p$ if we know q such that $(1+\delta)^q \leq v < (1+\delta)^{p+q}$. At height h, we work with a precision of $(1+\delta)^h$. As we compute the table for $u \in \mathcal{T}$ at height $h+1$ from the tables of its children at height h, the precision changes to $(1+\delta)^{h+1}$. This happens since the sum of two or more numbers that are known individually within a precision of $(1+\delta)^h$ is known only within a precision of $(1+\delta)^{h+1}$. (See footnote[4] for more explanation.) At the root, the final precision of the numbers is $(1+\delta)^{a \log n} = O(1)$.

We hope this difficult algorithm gives enough evidence why allowing overlap is better: the approximation algorithm for the overlapping case has much lower ratio and is much simpler.

4 Other conductance clustering problems

4.1 The MaxOverlapUnbound Problem

For MaxOverlapUnbound problem, since we should cover all the nodes, and we want to minimize the maximum conductance of the clusters, we can find the cut with the minimum conductance around each vertex $v \in V(G)$. The output of the algorithm is then the union of all cuts around all vertices. As a result, in order to solve the MaxOverlapUnbound problem, it is sufficient to solve the following problem: Given a bound B, a graph $G(V,E)$ and a vertex $v \in V(G)$, find a cut $(S, \overline{S})$ with minimum conductance $\phi(S)$ such that $v \in S$, and $\mathrm{vol}(S) \leq B$. We give a logarithmic approximation for this problem. In this algorithm, we use a recent result of Räcke [31] which is similar to the results [24].

Theorem 4.1 (Räcke [31]) *For any graph $G = (V,E)$, there exists a convex combination of tree decompositions T_i defined by multipliers λ_i with $\sum_i \lambda_i = 1$ and a constant C such that the following holds: for any $S \subset V$, (i) for any tree T_i, we have $w_G(\delta(S)) \leq w_{T_i}(S, \overline{S})$, and (ii) $\frac{1}{C \log n} \sum_i \lambda_i w_{T_i}(S, \overline{S}) \leq w_G(\delta(S))$.*

The above theorem implies that if we design an exact algorithm for our problem on trees, then using Racke's result, we can design a logarithmic approximation for our problem on all graphs. Now, we present a PTAS for the following problem P_1 on trees: Given a bound B, a graph G, and a vertex $v \in V$, find a cut $(S, \overline{S})$ with minimum conductance for which $v \in S$ and $\mathrm{vol}(S) \leq B$. In order to solve this problem, we first give a dynamic programming solution to a problem P_2 as

[4] If $(1+\delta)^{q_1} \leq a < (1+\delta)^{h+q_1}$ and $(1+\delta)^{q_2} \leq b < (1+\delta)^{h+q_2}$, then $(1+\delta)^q \leq a+b < (1+\delta)^{h+1+q}$ where $(1+\delta)^q \leq (1+\delta)^{q_1} + (1+\delta)^{q_2} < (1+\delta)^{q+1}$.

follows: Let T be a decomposition tree with edge weights w_e and polynomially-bounded integer leaf weights w_v. For each leaf node v, let $l_v \in \{0,1\}$ be the label of v: $l_v = 0$, we say that v is unsaturated, and $l_v = 1$, we say that v is saturated. Given the above input, problem P_2 is as follows: for any node u of the tree and any two numbers A and C, solve the following two problems: (i) find the minimum cut $(S, \overline{S})$ such that $u \in S$, the total leaf weight in S is A, and the total weight of unsaturated leafs in S is C, and (ii) find the minimum cut $(S, \overline{S})$ such that $u \notin S$, the total leaf weight in S is A, and the total weight of unsaturated leafs in S is C.

We call the minimum cut value in the former and the later cases $X(u, A, C)$ and $Y(u, A, C)$ respectively. The base case of the dynamic program is for small values of A and C, i.e, $0 \leq A, C \leq 1$. For a node u with two children nodes u_1 and u_2, in order to compute compute $X(u, A, C)$ and $Y(u, A, C)$, we consider all possible ways of decomposing A and C to four numbers A_1, A_2, and C_1, C_2 such that $A = A_1 + A_2$ and $C = C_1 + C_2$, and write the following recurrence relation:

$$\begin{aligned} X(u, A, C) &= \min(X(u_1, A_1, C_1) + X(u_2, A_2, C_2), \\ &\quad Y(u_1, A_1, C_1) + w_{u_1 u} + X(u_2, A_2, C_2), \\ &\quad Y(u_1, A_1, C_1) + w_{u_1 u} + Y(u_2, A_2, C_2) + w_{u_2 u}, \\ &\quad X(u_1, A_1, C_1) + Y(u_2, A_2, C_2) + w_{u_2 u}) \end{aligned}$$

and

$$\begin{aligned} Y(u, A, C) &= \min(X(u_1, A_1, C_1) + X(u_2, A_2, C_2) + w_{u_1 u} + w_{u_2 u}, \\ &\quad Y(u_1, A_1, C_1) + X(u_2, A_2, C_2) + w_{u_2 u}, \\ &\quad Y(u_1, A_1, C_1) + Y(u_2, A_2, C_2), \\ &\quad X(u_1, A_1, C_1) + Y(u_2, A_2, C_2) + w_{u_1 u}) \end{aligned}$$

In both the cases, the minimum is taken over all A_1, A_2, C_1, C_2 such that $A_1 + A_2 = A, C_1 + C_2 = C$.

The above dynamic program solves problem P_2. Now, in order to solve problem P_1, we first round all the weight of leafs of weight less than $\frac{\epsilon B}{n}$ for a small constant ϵ to zero. This rounding does not change the volume of the solution by a factor more than $1 + \epsilon$ as the total weight of such leaves is less than ϵB. Now, we can assume that the ratio between the maximum and minimum weight of leaves is at most $\frac{n}{\epsilon}$, since the leaves of weight more than B are not in the optimal solution. Now, we round all the remaining weights to a multiple of $\frac{\epsilon B}{n}$ which is smaller than the original weight of the leaf. This operation changes the total volume of clusters by at most a $1 + \epsilon$ factor. After performing the above rounding operation, the weight of the leaves are all in set $\{\frac{\epsilon}{n}B, \frac{2\epsilon}{n}B, \ldots, \frac{(n-1)\epsilon}{n}B, \epsilon B\}$, and thus by the right scaling, we can assume that weights are polynomially-bounded integers (as assumed in problem P_2). Now, we can solve problem P_2

on the new instance (with rounded weights) using the above dynamic program, and for a node u output the cut with minimum value among all min cuts corresponding to $X(u, A, C)$ for any two numbers $A, C \leq \frac{3}{2}B$. Because of the rounding method discussed above, this is a PTAS for problem P_1 on trees. This PTAS on trees and Racke's result imply a logarithmic approximation for the MaxOverlapUnbound problem.

4.2 The MaxOverlapBound Problem

To get a logarithmic approximation for the MaxOverlapBound problem, we again use Racke's result and reduce the problem to a problem on decomposition trees. For decomposition trees, we can design a PTAS using the dynamic program solution to problem P_2 described above. In fact, we run a set cover-type greedy algorithm and use the dynamic program as a procedure to implement the greedy set cover algorithm. In the following, we first present the algorithm, and then show that it can be implemented in polynomial time with a desired approximation factor. Given a decomposition tree T, we run the following set-cover-type algorithm ALG:

1. Guess the value of optimal solution OPT (i.e, try the following values for OPT: $\frac{\text{vol}(V(G))}{2^i}$ for $0 \leq i \leq \log \text{vol}(V(G))$.); and let $t = 0$.
2. While $\cup_{1 \leq i \leq t} S_i \neq V(G)$ do
   (a) Find a set $S_t \subseteq V(G)$ with the conductance of at most OPT which maximizes the total weight of uncovered nodes, i.e, the total weight of nodes in the set $S_t \cap (V(G) \setminus (\cup_{1 \leq i \leq t-1} S_i))$; and set $t := t + 1$.
3. Output $S_1, S_2, \ldots, S_t$.

To show that this algorithm can be implemented in polynomial time, we show a PTAS implementation of Line 2a of this algorithm. Similar to the rounding method used in solving problem P_1, we first round the weights of the leaves to polynomially-bounded integers. We call a node u saturated if u is already covered by set of clusters $S_1, \ldots, S_{t-1}$, i.e, $u \in \cup_{1 \leq i \leq t-1} S_i$. Now, we use the dynamic program for problem P_3 as follows: for each total weight $A \leq \frac{3}{2}B$ and each total weight of unsaturated nodes C, we find a cluster with the minimum cut value cutvalue(A, C). Among all these cuts with conductance $\frac{\text{cutvalue}(A,C)}{C} \leq$ OPT, we choose the cut the maximum C. This cut is the desired cut in Line 2a of Algorithm ALG.

We now prove the performance of Algorithm ALG for MaxOverlapBound. Let $U_1, U_2, \ldots, U_k (k \leq K)$ be optimal clusters for the MaxOverlapBound on a tree T. Also, consider a family $\mathcal{F}$ of subsets

of nodes of T corresponding to all clusters of conductance at most OPT each with a unit cost 1. Consider a set cover instance over this set system $\mathcal{F}$, i.e., a family of clusters with minimum cost (or minimum number of clusters) that covers all the elements (i.e., nodes of the graph). Since the conductance of all sets $U_1, \ldots, U_k$ is at most OPT, one can cover all nodes of the graph, using at most k sets in $\mathcal{F}$, and thus the cost of the optimal solution is at most k. While running algorithm ALG with a guess O for OPT such that OPT $\leq O \leq$ 2OPT, the algorithm is equivalent to the greedy set cover algorithm on the set cover instance $\mathcal{F}$. Therefore, the approximation factor of this algorithm is $\log n$ where n is the number of nodes, and as a result, this algorithm find at most $k \log n \leq K \log n$ clusters with min-max conductance of at most $(1+\epsilon)$OPT over tree T. Using Racke's result [31] along with this algorithm on trees, we get the desired algorithm for MaxOverlapBound with $O(\log n)$ approximation for the conductance and $K \log(n)$ clusters.

5 *Min-Sum* Conductance Minimization

For *min-sum* conductance optimization problems, we show that the optimum solution consists of clusters of volume $O(B)$, and thus reduce these problems to the the minimum balanced multi-cut optimization problems. Therefore, using Lemma2.2 and the known $O(\log n)$ approximation algorithms for the minimum balanced multi-cut problems [31], we get logarithmic approximation algorithms for SumOverlapBound.

In section 2, we showed that the overlapping and non-overlapping problems are equivalent up to a constant factor, therefore we design logarithmic approximation for non-overlapping clustering variants and it implies a logarithmic approximation for overlapping variants as well. In order to solve SumNonoverlapUnbound, we first observe that there exists a clustering of total cost less than or equal to the optimal solution in which all clusters have volume between $\frac{B}{2}$ and $\frac{3}{2}B$. The reason is that in any optimal solution, if any cluster has volume less than $\frac{B}{2}$, we can substitute this cluster and another arbitrary cluster and only decrease the sum of the conductance of the clusters. As a result, the cut minimization problem and the conductance minimization problem are equivalent for this objective function up to a constant factor. The minimum cut balanced partitioning problem admits a polynomial-time $O(\log n)$-approximation algorithm [18], and thus the SumNonoverlapUnbound also admits a logarithmic approximation algorithm. Now, in order to solve SumNonoverlapBound, we note that since in the solution of SumNonoverlapUnbound, the volume of clusters is between $\frac{B}{2}$ and $\frac{3B}{2}$, the number of clusters is at most $O(\frac{\text{vol}(V(G))}{B})$. Moreover, for SumNonoverlapBound to have a feasible solution, we should have $K \geq \frac{\text{vol}(V(G))}{B}$. As a result, by running the logarithmic approximation algorithm SumNonoverlapUnbound, the output has at most $O(K)$ clusters, and its solution is a logarithmic approximation to SumNonoverlapBound as desired.

6 Conclusion

In this paper, we set out to examine the advantage in allowing overlap in clustering from a theoretical point of view. The problem chosen is the popular one of minimizing conductance. We introduced several overlapping and non-overlapping clustering problems, and presented a first set of approximation algorithms for them. In one case (min-max) the intuition that overlap clustering should be better was proved true as the optimum may be much smaller and the approximation much easier. In the sum version, contrary to our intuition, overlapping does not help much in terms of the approximability of the problem. Many interesting problems remain open. Other than improving the approximation algorithms, fast combinatorial algorithms for the minimum conductance problems, and studying overlapping clustering for other previously-studied non-overlapping clustering problems are interesting subjects of study.

Acknowledgement. We thank David Gleich for useful discussions, for validating the importance of our model and for conducting some initial experimental study.

References

[1] A. Agarwal, N. Alon, and M. Charikar. Improved approximation for directed cut problems. In *STOC*, pages 671–680, 2007.

[2] R. Andersen, F. R. K. Chung, and K. J. Lang. Local graph partitioning using pagerank vectors. In *FOCS*, pages 475–486, 2006.

[3] R. Andersen, D. Gleich, and V. Mirrokni. Overlapping clustering for distributed computation. In *ACM Conference on Web search and Data Mining*, 2012.

[4] S. Arora, R. Ge, S. Sachdeva, and G. Schoenebeck. Finding overlapping communities in social networks: Toward a rigorous approach. In *ACM EC*, 2012.

[5] S. Arora, E. Hazan, and S. Kale. O(sqrt(log(n)) approximation to sparsest cut in $\tilde{o}(n^2)$ time. *SIAM J. Comput.*, 39(5):1748–1771, 2010.

[6] S. Arora, J. R. Lee, and A. Naor. Euclidean distortion and the sparsest cut. In *STOC '05: Proceedings of the thirty-seventh annual ACM symposium on Theory of computing*, pages 553–562, New York, NY, USA, 2005. ACM Press.

[7] S. Arora, S. Rao, and U. V. Vazirani. Expander flows, geometric embeddings and graph partitioning. In *STOC*, pages 222–231, 2004.

[8] M. Balcan, C. Borgs, M. Braverman, J. T. Chayes, and S. Teng. I like her more than you: Self-determined communities. *CoRR*, abs/1201.4899, 2012.

[9] N. Bansal, U. Feige, R. Krauthgamer, K. Makarychev, V. Nagarajan, J. Naor, and R. Schwartz. Min-max graph partitioning and small set expansion. In *FOCS*, pages 17–26, 2011.

[10] A. Blum and S. Chawla. Learning from labeled and unlabeled data using graph mincuts. In *ICML*, pages 19–26, 2001.

[11] U. Brandes, M. Gaertler, and D. Wagner. Engineering graph clustering: Models and experimental evaluation. *ACM J. Experimental Algorithmics*, 1(1), 2007.

[12] G. Călinescu, H. J. Karloff, and Y. Rabani. An improved approximation algorithm for multiway cut. *J. Comput. Syst. Sci.*, 60(3):564–574, 2000.

[13] S. Chawla, A. Gupta, and H. Räcke. Embeddings of negative-type metrics and an improved approximation to generalized sparsest cut. In *SODA '05: Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 102–111, Philadelphia, PA, USA, 2005. Society for Industrial and Applied Mathematics.

[14] J. Cheriyan, H. Karloff, and Y. Rabani. Approximating directed multicuts. *Combinatorica*, 25(3):251–269, 2005.

[15] J. Chuzhoy and S. Khanna. Hardness of cut problems in directed graphs. In *STOC '06: Proceedings of the thirty-eighth annual ACM symposium on Theory of computing*, pages 527–536, New York, NY, USA, 2006. ACM Press.

[16] J. Chuzhoy and S. Khanna. Polynomial flow-cut gaps and hardness of directed cut problems. *J. ACM*, 56(2), 2009.

[17] I. Dinur and S. Safra. The importance of being biased. In *Symposium on Theory of Computing*, pages 33–42, 2002.

[18] G. Even, J. Naor, S. Rao, and B. Schieber. Fast approximate graph partitioning algorithms. *SIAM J. Comput.*, 28(6):2187–2214, 1999.

[19] U. Feige, D. Peleg, and G. Kortsarz. The dense k -subgraph problem. *Algorithmica*, 29(3):410–421, 2001.

[20] N. Garg, V. V. Vazirani, and M. Yannakakis. Approximate max-flow min-(multi)cut theorems and their applications. *SIAM Journal on Computing*, 25(2):235–251, 1996.

[21] N. Garg, V. V. Vazirani, and M. Yannakakis. Primal-dual approximation algorithms for integral flow and multicut in trees. *Algorithmica*, 18(1):3–20, 1997.

[22] U. Gargi, W. Lu, V. Mirrokni, and S. Yoon. Large-scale community detection on youtube. In *ICWSM*, 2011.

[23] A. Gupta. Improved results for directed multicut. In *Symposium on Discrete Algorithms*, pages 454–455, 2003.

[24] C. Harrelson, K. Hildrum, and S. Rao. A polynomial-time tree decomposition to minimize congestion. In *SPAA*, pages 34–43, 2003.

[25] R. Khandekar, S. Rao, and U. V. Vazirani. Graph partitioning using single commodity flows. In *STOC*, pages 385–390, 2006.

[26] Y. Kortsarts, G. Kortsarz, and Z. Nutov. Greedy approximation algorithms for directed multicuts. *Networks*, 45(4):214–217, 2005.

[27] J. R. Lee. On distance scales, embeddings, and efficient relaxations of the cut cone. In *SODA '05: Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 92–101, Philadelphia, PA, USA, 2005. Society for Industrial and Applied Mathematics.

[28] R. Lepère and C. Rapine. An asymptotic o(ln rho/ln ln rho)-approximation algorithm for the scheduling problem with duplication on large communication delay graphs. In *STACS*, pages 154–165, 2002.

[29] N. Mishra, R. Schreiber, I. Stanton, and R. E. Tarjan. Clustering social networks. In *WAW*, pages 56–67, 2007.

[30] H. Räcke. Minimizing congestion in general networks. In *FOCS*, pages 43–52, 2002.

[31] H. Räcke. Optimal hierarchical decompositions for congestion minimization in networks. In *STOC*, pages 255–264, 2008.

[32] H. Räcke. Optimal hierarchical decompositions for congestion minimization in networks. In *STOC*, pages 255–264, 2008.

[33] T. Sahai, A. Speranzon, and A. Banaszuk. Hearing the clusters of a graph: A distributed algorithm. *Automatica*, 48(1):15–24, 2012.

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

[35] D. Shmoys. Cut problems and their applications to divide-andconquer, 1996.

[36] A. P. Streich, M. Frank, D. Basin, and J. M. Buhmann. Multi-assignment clustering for boolean data. In *ICML*, 2009.