A Near-Linear Time Approximation Algorithm for Beyond-Worst-Case Graph Clustering

Vincent Cohen-Addad [*1] **Tommaso d'Orsi** [*1 2] **Aida Mousavifar** [*1]

Abstract

We consider the semi-random graph model of (Makarychev et al., 2012), where, given a random bipartite graph with α edges and an unknown bipartition (A, B) of the vertex set, an adversary can add arbitrary edges inside each community and remove arbitrary edges from the cut (A, B) (i.e. all adversarial changes are *monotone* with respect to the bipartition). For this model, a polynomial time algorithm is known to approximate the Balanced Cut problem up to value $O(\alpha)$ (Makarychev et al., 2012) as long as the cut (A, B) has size $\Omega(\alpha)$. However, it consists of slow subroutines requiring optimal solutions for logarithmically many semidefinite programs. We study the fine-grained complexity of the problem and present the first near-linear time algorithm that achieves similar performances to that of (Makarychev et al., 2012). Our algorithm runs in time $O(|V(G)|^{1+o(1)} + |E(G)|^{1+o(1)})$ and finds a balanced cut of value $O(\alpha)$. Our approach appears easily extendible to related problem, such as Sparsest Cut, and also yields an near-linear time $O(1)$-approximation to Dagupta's objective function for hierarchical clustering (Dasgupta, 2016) for the semi-random hierarchical stochastic block model inputs of (Cohen-Addad et al., 2019).

1. Introduction

Graph clustering and partitioning problems are central in combinatorial optimization. Their study has led to a large variety of key results, leading to new fundamental ideas and impactful practical outcomes. The sparsest cut and balanced cut problems are iconic examples: On the one hand, they have served as a testbed for designing new breakthrough algorithmic techniques, from the seminal paper of Leighton and Rao (Leighton & Rao, 1988) up to the results of Arora, Rao, and Vazirani (Arora et al., 2004) and Sherman (Sherman, 2009). On the other hand, they are models for graph partitioning problems in various data mining and unsupervised machine learning applications and have thus inspired widely-used heuristics in more applied fields.

Beyond worst-case instances A frustrating gap exists between the impressive theoretical results obtained over the last three decades and the success of heuristics used in practice. While poly-logarithmic approximation algorithms have been developed for balanced cut and sparsest cut (and related problem such as minimum bisection (Räcke, 2008), multicut (Garg et al., 1996), min uncut (Goemans & Williamson, 1995; Agarwal et al., 2005)), the algorithm design community has had little success in obtaining constant factor approximation algorithms for these problems. In fact, the Unique Games Conjecture even suggests that such bounds may be very hard to obtain (Khot et al., 2007; Khot & Vishnoi, 2015; Raghavendra, 2008; Raghavendra et al., 2012). Thus, to be able to show good approximation bounds and design algorithms that are tailored to real-world instances, one must shift the focus from the worst-case to the so called *beyond-worst-case* complexity of the problems.

This conclusion has seeded a long line of work aimed at modeling average instances encountered in practice and designing algorithms for these models (Dyer & Frieze, 1986; Bui et al., 1987; Boppana, 1987; Feige & Kilian, 2001; McSherry, 2001) (or analyzing existing algorithms in these models (Jerrum & Sorkin, 1993; Dimitriou & Impagliazzo, 1998; Bilu & Linial, 2012b)). For the model to be relevant it should forbid pathological instances that are extremely unlikely in practice while capturing the essence of the real-world instances without oversimplifying them.

While there has been a significant amount of work on inference in random and semi-random graph models, the work of Makarychev, Makarychev and Vijayaraghavan (Makarychev et al., 2012) is among the first to analyze the approximability and complexity of the graph partitioning objectives mentioned above for extremely general families of semi-random graphs. In their settings, the input is generated from

[*]Equal contribution [1]Google Research [2]BIDSA, Bocconi. Correspondence to: Tommaso d'Orsi <tommaso.dorsi@unibocconi.it>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024.

a distribution over graphs that exhibit a cluster structure. Concretely, the graph consists of two communities and a planted random cut between the communities, the adversary can modify the graph in agreement with the cluster structure by arbitrarily adding edges within the communites and / or sparsifying the random cut across communities,[1] see Model 1.1 for a precise definition. In this context, the goal is not to recover the underlying cluster structure – which may be information-theoretically impossible – but rather to provide a good approximation to the cut objectives.

The motivation for studying such models is the following. In practice, the graphs we aim at clustering have an unknown underlying cluster structure that we would like to identify – and that's why we are running a clustering algorithm in the first place. In this context, on the one hand the intra-cluster topology may be very peculiar and so possibly adversarial (hence we would like to let the adversary freely choose the intra-cluster topology[2]), on the other hand the inter-cluster topology is often more random, sometimes interpreted as noise between clusters and hence modeled as a random cut, see also the discussion and motivating examples provided in (Makarychev et al., 2012).

Of course, allowing the adversary to make the planted cut denser – and by doing so to smooth out the underlying cluster structure – would bring us back to the worst-case setting; the semi-random model proposed above is thus a step in between.

Hence, with the idea of bridging the gap between worst-case complexity of the problems and real-world instances, Makarychev, Makarychev, Vijayaraghavan (Makarychev et al., 2012) developed a general algorithmic framework for graph partitioning problems in the above semi-random instances which achieves an $O(1)$-approximation algorithm (for a wide array of parameters) for balanced cut and sparsest cut and related problems such as multicut, min uncut and small set expansion.

While the result of (Makarychev et al., 2012) is close to optimal in the sense that it achieves an $O(1)$-approximation for several classic graph partitioning problems and a wide range of parameters, it relies on an heavy machinery that requires to iteratively solve multiple semi-definite programs. In fact the running time is not stated in the paper and seem to require $\Omega(n^3)$ time for the rounding on top of the time it takes to obtain optimal solutions to polylogarithmically many semi-definite programs with more than $\Omega(n^3)$ constraints.[3]

We initiate the study of the *fine-grained* complexity of the problem and ask: *How fast can we solve beyond-worst-case instances (involving semi-random perturbations)?*

1.1. Results

Before providing our main theorem, we introduce the model of interest.

Model 1.1 (Random cut with monotone perturbations)**.** We consider graphs over n vertices generated through the following process. Let $a \in (0, 1/2)$, $\eta(n) \in (0, 1)$:

(i) The adversary partitions $[n]$ into sets A, B satisfying $|A|, |B| \geqslant an$.

(ii) Each edge between A and B is drawn randomly and independently with probability η.

(iii) The adversary arbitrarily adds edges within A and within B.

(iv) The adversary arbitrarily removes edges between A and B.

Our main result is an algorithm that, given an instance of Model 1.1 with a $\Omega(n^2 \cdot \eta)$-sized (A, B) cut, returns a $O(1)$-approximation in almost linear time.[4]

Theorem 1.2. *Let G be a graph over n vertices generated through Model 1.1 with parameters $a > 0, \eta \geqslant \Omega(\frac{(\log n)^2 \cdot (\log\log n)^2}{n})$. There exists an algorithm that on input G, with probability $1 - o(1)$, outputs an $\Omega(a)$-balanced cut of value at most $O(n^2 \cdot \eta)$, namely a cut where each side has size at least $\Omega(a \cdot n)$.*

Moreover, the algorithm runs in time $O\left(|V(G)|^{1+o(1)} + |E(G)|^{1+o(1)}\right)$.

Theorem 1.2 is a significant step toward bridging the gap between the theoretically-oriented work of (Makarychev et al., 2012) and the practical motivation behind semi-random models. The error guarantees of the underlying algorithm match those of (Makarychev et al., 2012), but the running time is nearly linear. Despite the fact that further steps remains to be taken to provide algorithmic solutions that both matches the theoretical guarantees of (Makarychev et al., 2012) and whose running time is competitive with state-of-the-art Bisection heuristics, our algorithm is a *first* example that general beyond-worst-case graph clustering can be done in near linear time.

Finally, we believe that understanding the fine-grained complexity of balanced cut and related problems beyond-the-

[1] These are often times referred to as *monotone* perturbations. Such perturbations may have surprising effects on the statistical and computational aspects of the problem. For instance see (Moitra et al., 2016; Liu & Moitra, 2022).

[2] We remark this model is significantly more general than the stochastic block model, see Section 1.2

[3] We point out that the algorithm *requires* an *actual* feasible solution with nearly optimal objective value and not a rounded solution.

[4] We write $o(1)$ to denote real-valued functions tending to zero as n grows.

worst case is an important line of work and the techniques presented here could lead to further improvements for other related problems for which the beyond-worst-case analysis has been studied (e.g.: Bilu-Linial stability for multicut (Bilu & Linial, 2012a; Angelidakis et al., 2017)).

Generalizations Our approach appears to also be easily extendable to other graph problems. As a concrete example, we consider the *semi-random hierarchical stochastic block model* (henceforth HSM) of (Cohen-Addad et al., 2019). In (Cohen-Addad et al., 2019), the authors studied the celebrated objective function for hierarchical clustering introduced by Dasgupta (Dasgupta, 2016) and investigate how well it can be approximated beyond-the-worst-case. Assuming the Small Set Expansion hypothesis (Raghavendra & Steurer, 2010), the problem cannot be approximated within any constant factor. The authors thus introduce a generative model for hierarchical clustering inputs called the *hierarchical stochastic block model* that naturally generalizes the classic stochastic block model, and show that one can approximate Dasgupta's objective up to a constant factor in that model and under semi-random perturbation (the precise definition of the model can be found in Appendix D). In this paper, we significantly improve the complexity of the algorithm of (Cohen-Addad et al., 2019).

Theorem 1.3. *Let G be a graph generated from the HSM (Definition D.4) with $p_{min} = \Omega\left(\log n/n^{2/3}\right)$. Then, there exists a randomized algorithm that runs in time $O\left(|V(G)|^{1+o(1)} + |E(G)|^{1+o(1)}\right)$ with probability $1 - o(1)$ outputs a tree T such that*

$$cost(T; G) = O(OPT(\bar{G})), \tag{1}$$

where $OPT(\bar{G})$ denotes the value of the optimal tree for $\bar{G}$ and we note that $OPT(\bar{G}) = cost(\tilde{T}; \bar{G})$, where $\tilde{T}$ is the generating tree. Furthermore, the above holds even in the semi-random case, i.e., when an adversary is allowed to remove any subset of the edges from G.

1.2. Related Research

There has been extensive research on graph partitioning problems for random and semi-random models. Perhaps the most extensively studied example is the stochastic block model (see (Abbe, 2017) for a broad overview). In its simplest form, the model describes graphs where both the inter-community and the intra-community topologies are random. That is the graph is randomly partitioned into two subsets (A, B) of the same size such that every edge between the set A and set B exists with probability η, and edges inside communities A, and B exists with probability $\mu \geqslant \eta$.[5] Many algorithms are known to succesfully recover the partition for typical instances of the model (Decelle et al., 2011; Massoulié, 2014; Mossel et al., 2015; Hopkins & Steurer, 2017; Mossel et al., 2018). In recent years, an ongoing line of work has aimed to extend these algorithmic techniques to more general semi-random models (Feige & Kilian, 2001; Moitra et al., 2016; Montanari & Sen, 2016; Ding et al., 2022; Hua et al., 2023), first by introducing monotone perturbations (Feige & Kilian, 2001; Moitra et al., 2016; Fei & Chen, 2019; Liu & Moitra, 2022) (a perturbation is monotone with respect to the bipartition (A, B) if it adds edges inside the comunities or remove edges accross communities) and then by allowing a small but constant fraction of adversarially chosen edge (Ding et al., 2022) or vertex (Liu & Moitra, 2022; Hua et al., 2023) perturbations. These results still crucially rely on the randomness of the intra-cluster topology and thus cannot work in the significantly more general context of Model 1.1, where the structure inside A and B is arbitrarily and not random. (We remark that random settings with $0 < \mu \leqslant \eta$ have been investigated in the statistical physics literature.[6] These results also cannot be used in the presence of monotone perturbations.)

Model 1.1 was extended in (Makarychev et al., 2014), where the same set of authors introduced the so-called PIE model. Here the main assumption concerning the edges across the (A, B) partition is that they were sampled from a permutationally invariant distribution (w.r.t to edges). Their error guarantees are comparable to those of (Makarychev et al., 2012) in the denser regime $\eta \geqslant O(\frac{\text{polylog}(n)}{n})$. We leave it as an open question to extend Theorem 1.2 to this model.

Less general semi-random models, in which adversarial perturbations are applied before sampling the random edges, have also been studied. Interestingly, for these significantly weaker adversaries, spectral algorithms have been shown to achieve nearly optimal guarantees (Chierichetti et al., 2022). However, as already mentioned, these algorithms are known to be fragile to perturbations such as in Model 1.1 and thus cannot be expected to lead to error guarantees comparable to those in Theorem 1.2.

Further recent work (Peng, 2020) developed a sublinear robust algorithm for local reconstruction of noisy $(k, \phi_{\text{in}}, \phi_{\text{out}})$-clusterable graphs. Such graphs consists of k expanders with inner conductance at least ϕ_{in} and outer conductance at most ϕ_{out} and an adversary is allowed to modify at most ε fraction of edges within clusters. This result holds under the assumption $\phi_{\text{out}} \leqslant \frac{\varepsilon \cdot \phi_{\text{in}}}{\log n \cdot k^{O(1)}}$.

[5] We remark that from both a computational and a statistical point of view, sharp phase transitions appear depending on the relation between the expected average degree and the community bias. We omit a detailed discussion and refer the interested reader to the aforementioned survey.

[6] The model is usually referred to as the *antiferromagnetic model*.

2. Techniques

We present here the main ideas contained in the proof of Theorem 1.2. Throughout the section let G be a random graph sampled through steps (i) and (ii) of Model 1.1 and let G° be the resulting graph after steps (iii) and (iv). We let $\alpha \leqslant (1+o(1)) \cdot n^2 \cdot \eta$ be the number of edges in G and thus an upper bound on the optimal balanced cut in G° with high probability.

Slow algorithms for balanced cut in the semi-random model In order to present our techniques for obtaining a near linear time algorithm with constant approximation factor for Model 1.1, it is necessary to first understand how the known *slow* algorithm of (Makarychev et al., 2012) works. Consider the random graph G and let $v_1, \ldots, v_n \in \mathbb{R}^n$ be the embedding given by the returned SDP solution, where v_i corresponds to the embedding of the i-th vertex of G.[7] The algorithm of (Makarychev et al., 2012) is an iterative procedure that cycles over two subroutines: first, the algorithm solves the canonical balanced cut SDP as in (Arora et al., 2004); second, the algorithm *carefully* removes clusters of vertices that are particularly *close* to each other in the embedding given by the found optimal SDP solution.

Concretely, the latter step identifies so-called (δ, n)-*heavy* vertices, namely vertices i such that its embedding v_i in the SDP solution is at distance at most δ from at least $10\delta^2 n$ embeddings of other vertices in the SDP solution[8]. Then, the algorithm *carves out* a ball of radius δ: It creates a cluster containing i and all the vertices j s.t. v_j is at distance at most δ from v_i. Here δ is a parameter chosen appropriately. At the end of this process, the algorithm has removed a set $H_\delta \subseteq V(G)$ of vertices from the instance.

The crucial observation here is that, in *every* feasible embedding of the random graph G, if the random cut is dense enough, then when *restricted* to the non-heavy vertices, it will satisfy a one-sided Chebyshev-like inequality of the form:

$$\mathbb{P}_{ij \overset{u.a.r.}{\sim} E(G \setminus H_\delta)} \left(\|v_i - v_j\|^2 \leqslant \delta \right) \leqslant 1/\delta^2 \,.$$

That is, with high probability only a $O(\delta^2)$-fraction of the edges between non-heavy vertices is shorter than δ in the embedding. This property is called *geometric expansion*.

Now the crux of the argument is that, given a feasible embedding $v_1, \ldots, v_n \in \mathbb{R}^n$ of G, if by removing heavy vertices we don't cut more than $O(\alpha)$ edges, then the geometric expansion property guarantees that the minimum balanced cut in the *remaining* graph has cardinality at most $O(\delta^2 \cdot \alpha)$. Thus after several iterations of the algorithm we have decreased the value of the minimum balanced cut by at least a $O(1/\sqrt{\log n})$ factor (and in fact, we will need to decrease it by a $1/\log n$ factor in order to use the algorithm of (Sherman, 2009) in near linear time) and so a simple application of the SDP rounding of (Arora et al., 2004) returns now a cut of optimal value α. Importantly, the geometric expansion property is robust to monotone changes – namely to the changes that the adversary can make to the graph at the last two steps of the generative model (Model 1.1) and thus, the exact same reasoning applies for the graph G° as well.

Roadblocks to speeding up the algorithms via the matrix multiplicative weights framework While semidefinite programs are computationally expensive to solve, there is by now a rich literature on fast algorithms to approximately solve them (e.g. see (Arora & Kale, 2007; Sherman, 2009; Steurer, 2010), see Appendix A for a comprehensive description). These results rely on the matrix multiplicative weight method (henceforth MMW) (Arora & Kale, 2007).[9] The framework aims at obtaining a "feasible enough" solution to the SDP so that the desired rounding argument works out. On a high level, the approach is based on the following steps: *(i)* find an assignment of the program variables that is only approximately feasible, in the sense that only a subset of constraints is approximately satisfied, *(ii)* round this infeasible solution into a feasible, integral solution. The key underlying idea is that, for many problems, if the subset of constraints that gets satisfied is chosen carefully, then the rounding algorithm works even though the starting assignment is far from being feasible. The running time improvement is obtained by designing an ORACLE algorithm that efficiently answers **yes**, if the candidate solution is feasible enough, or otherwise answers **no** and exhibits constraints that are violated by the current solution. These will then be used to pick the direction of movement in the underlying mirror descend algorithm. The running time of the oracle depends on the so-called oracle's *width* (see Appendix A) and thus the challenge is usually to design oracles of bounded width.

In the context of balanced cut for arbitrary graphs, this approach has been extremely successful, leading to an $O(\sqrt{\log n}/\varepsilon)$ approximation algorithm running in time $O(n^{1+\varepsilon} \log n + m)$ when combining (Sherman, 2009) and maximum flow algorithm of (Chen et al., 2022).

With respect to our settings, a natural question to ask is whether this matrix multiplicative framework may be used at each iteration of the slow algorithm above to improve its running time. While this intuition is –in principle– correct, to obtain significant running time speed ups, additional fundamental challenges need to be solved. First, the heavy

[7] See Section 4 for a definition of the program.

[8] Notice that a ball of radius 2δ centered at any heavy vertex v contains at least $\delta^2 n$ vertices.

[9] This can be seen as an application of the mirror descent algorithm with the von Neumann negative entropy as the chosen mirror map.

vertices removal procedure, that requires to find out all the heavy vertices of the graph – or in other words, all the particularly dense balls in the SDP solution– is slow. Second and most important, this subroutine crucially relies on having access to an *optimal* SDP solution. Hence the procedure cannot work with the infeasible solutions computed by the MMW framework.

To overcome these obstacles we need to deviate from the canonical matrix multiplicative weights paradigm.

Approximate heavy vertices removal Our first improvement, thus, consists of designing a faster algorithm to replace the subroutine that identifies all the dense balls of the optimal SDP solution. Identifying dense balls in high-dimensional Euclidean spaces (say of dimension $\Omega(\varepsilon^{-2}\log n)$) is a well studied problem. We make use of subsampling techniques to *approximately* solve this problem. Namely, our procedure will recover all the heavy vertices but may yield false positives, namely balls that are almost dense – up to a constant factor away from the target density. Concretely, we ask for balls that contain at least $10\delta^2 n$ vertices but are of radius $\sqrt{2}\delta$ instead of δ. While we can achieve this in time $\tilde{O}(|V(G)|)$, we now have to modify the next steps of the rounding to take the false positive into account in the rounding.

Main challenge: rounding and the matrix multiplicative weights framework The second challenge is more significant. Our idea is the introduction of a probabilistic oracle of small width that leverages the geometric expansion of the planted cut. Concretely, recall in the previous paragraph we introduced an approximate heavy vertex removal procedure with the property that, with constant probability, only a few edges will be cut *if* applied to a feasible solution of small objective value. It is important to notice that if the procedure cuts few edges even when applied to an infeasible solution of high objective value, we would still be satisfied since we would be making good progress in the graph partitioning at low cost. That is, even if a candidate solution is overall far from being feasible, it turns out to be sufficiently good for us if it is close to being feasible on the heavy vertices and their neighborhoods. We thus mainly have to deal with the case where too many edges get cut. The crux of the argument then is that if the procedure cuts too many edges, we can show how with reasonable probability there exists a hyperplane of *small width* separating our solution from the set of feasible solutions of small objective value, and moreover we can identify it efficiently. This means we can make progress and obtain a better solution through applying a step of the matrix multiplicative weights framework.

The intuition behind the **no**-case of this probabilistic oracle is that, *on average*, the probability that an edge with exactly one endpoint in these heavy balls is cut throughout the removal procedure must be larger than for feasible embeddings with small objective value. Indeed otherwise we could have expected our procedure to cut fewer edges. Thus we can conclude that with constant probability, either the current solution has significantly larger objective value, or several triangle inequalities must be violated at the same time and thus we can provide a feedback matrix of small width.

To ensure that our heavy vertices removal procedure would have indeed cut fewer edges if given a feasible solution, we repeat this process poly-logarithmically many times.

Remark 2.1 (On the minimum edge density η in the cut). As already briefly discussed, Theorem 1.2 requires $\eta \geqslant \Omega(\frac{(\log n)^2\cdot(\log\log n)^2}{n})$. In comparison (Makarychev et al., 2012) only requires a lower bound of $\Omega(\frac{(\sqrt{\log n}\cdot(\log\log n)^2}{n})$. This discrepancy comes from the fact that a near linear time budget only allows us to only obtain a $O(\log n)$ approximation to balanced cut using (Sherman, 2009; Chen et al., 2022). We offset these worse guarantees leveraging stronger geometric expansion properties, which may not hold for $\eta < O(\frac{(\log n)^2\cdot(\log\log n)^2}{n})$. What density is necessary for constant approximation to be possible, remains a fascinating open question.

2.1. Perspective

There is also a second perspective from which we may see Theorem 1.2. The last decade has seen tremendous advancements in the design of algorithms for inference problems that are robust to adversarial corruptions (among many we cite (d'Orsi et al., 2020; Ding et al., 2022; Liu & Moitra, 2022; Bakshi et al., 2022; Guruswami et al., 2022; Hua et al., 2023), see also the survey (Diakonikolas & Kane, 2019)). The emerging picture, which unfortunately appears hard to formalize, is that certain algorithmic techniques –such as semidefinite programming– appear robust while others –such as low-degree polynomials or local search– can be fooled by carefully chosen perturbations. In particular, the use of optimal solutions for semidefinite programs have been a fundamental tool behind these results. Unfortunately, the computational budget required to find such objects is often large, making comparable results hard to achieve in near linear time. Theorem 1.2 is a first significant example in which one can retain this *"robustness property"* while working with a near-linear time computational budget, hence providing a first step towards more practical robust algorithms.

3. Organization and notation

The rest of the paper is organized as follows. In Section 4 we present the algorithm. We provide necessary background on the matrix multiplicative framework and on other notions

used throughout the paper in Appendix A. We present our ORACLE in Appendix C, which combined with the results in Section 4 yields Theorem 1.2. We then study the hierarchical stochastic block in Appendix D.

We hide multiplicative factors *poly-logarithmic* in n using the notation $\tilde{O}(\cdot), \tilde{\Omega}(\cdot)$. Similarly, we hide absolute constant multiplicative factors using the standard notation $O(\cdot), \Omega(\cdot), \Theta(\cdot)$. Often times we use the letter C to denote universal constants independent of the parameters at play. Given a function $g : \mathbb{R} \to \mathbb{R}$, we write $o(g)$ for the set of real-valued functions f such that $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$. Similarly, we write $g \in \omega(f)$ if $f \in o(g)$. Throughout the paper, when we say "an algorithm runs in time $O(q)$" we mean that the number of basic arithmetic operations involved is $O(q)$. That is, we ignore bit complexity issues.

Vectors and matrices We use Id_n to denote the n-by-n dimensional matrix and $\mathbf{0}$ to denote the zero matrix. For matrices $A, B \in \mathbb{R}^{n\times n}$ we write $A \succeq B$ if $A - B$ is positive semidefinite. For a matrix M, we denote its eigenvalues by $\lambda_1(M), \ldots, \lambda_n(M)$; we simply write λ_i when the context is clear. We denote by $\|M\|$ the spectral norm of M. Let $\mathcal{S}_n \subset \mathbb{R}^n$ be the set of real symmetric n-by-n matrices and let $\Delta_n(r) := \{X \in \mathcal{S}_n \mid \operatorname{Tr} X \leqslant r, X \succeq 0\}$. For $X \in \mathcal{S}_n$, the matrix exponential is $\exp(X) = \sum_{i=0}^{\infty} \frac{X^i}{i!}$. We remark that $\exp(X)$ is positive semidefinite for all symmetric X as $\exp(X) = \left(\exp(\frac{1}{2}X)\right)^{\mathsf{T}} \exp(\frac{1}{2}X)$. For a vector $v \in \mathbb{R}^n$, we write $v \geqslant \mathbf{0}$ if all entries of v are non-negative. We use $\mathbb{S}^n \subseteq \mathbb{R}^n$ to denote the unit sphere.

Graphs We denote graphs with the notation $G(V, E)$. We use $V(G)$ to denote the set of vertices in G and similarly $E(G)$ to denote its set of edges. For a graph G we write L_G for the associated combinatorial Laplacian, which is a matrix with rows and columns indexed by the nodes of G such that $(L_G)_{ii} = \sum_{ij\in E(G)} 1$, i.e. the degree of node i, and for $i \neq j$ $(L_G)_{ij}$ is -1 if $ij \in E(G)$ and 0 otherwise. When the context is clear we drop the specification of G. Unless specified otherwise, we use n to denote $|V(G)|$. For a partition (A, B) of the vertices of G, we write $E(A, B) \subseteq E$ for the set of edges in the A-B cut. We say that a partition (A, B) is a-balanced if $|A| / |B| \geqslant a$ assuming $|A| \leqslant |B|$.

4. A fast algorithm for semi-random balanced cut

We present here our main theorem which implies Theorem 1.2. To solve the balanced cut problem we consider its basic SDP relaxation. Given a graph G, the relaxation for the a-balanced cut problem is:

$$\left\{\begin{array}{ll} \min \sum_{ij\in E} c_{ij} \|v_i - v_j\|^2 & \\ \|v_i\|^2 = 1 & \forall i \in [n] \\ \|v_i - v_j\|^2 + \|v_j - v_k\|^2 & \\ \qquad \geqslant \|v_i - v_k\|^2 & \forall i, j, k, \in [n] \\ \sum_{i,j\in[n]} \|v_i - v_j\|^2 \geqslant 4an^2 & \end{array}\right\} \tag{2}$$

We refer to the first constraint as the unit norm constraint, to the second as the triangle inequality constraint and to the third as the balance constraint. Equation (2) may be rewritten in its canonical form

$$\left\{\begin{array}{ll} \min \langle L, X\rangle & \\ X_{ii} = 1 & \forall i \in [n] \\ \langle T_p, X\rangle \geqslant 0 & \forall \text{ paths } p \text{ of length 2} \\ \langle K_V, X\rangle \geqslant 4an^2 & \end{array}\right\} \tag{3}$$

where L is the combinatorial Laplacian of the graph, K_S is the Laplacian of the complete graph over vertex set S and, for a path p, T_p is the difference between the Laplacian of p and the Laplacian of the single edge connecting its endpoints. Notice that $v_1, \ldots, v_n$ are the Gram vectors of X. To ease the reading we will sometimes use the vectors representation and others the matrix representation. We denote by α the optimal value for a given instance of Equation (3). In particular, we say a graph G has optimal cut α if minimum solutions to Equation (2) have objective value α. Notice that for graphs generated as in Model 1.1, with high probability we have $\alpha \leqslant (1 + o(1))n^2 \cdot \eta$.

Before stating the main theorem we require a couple of definitions concerning the embedding of graphs. These are based on (Makarychev et al., 2012).

Definition 4.1 (Heavy vertex)**.** Let $\delta, n, n' > 0$. Let $G(V, E)$ be a graph on n vertices and let X be the Gram matrix of an embedding $v_1, \ldots, v_n \in \mathbb{R}^n$ of G onto $\mathbb{R}^n$. A vertex $i \in V$ is said to be (δ, n')-heavy if

$$\left|\left\{j \in V(G) \;\middle|\; \|v_i - v_j\|^2 \leqslant \delta\right\}\right| \geqslant \delta^2 \cdot n'.$$

We denote the set of (δ, n')-heavy vertices in the embedding X by $H_{\delta,n'}(X, V)$. For a subset of vertices V' we let $H_{\delta,n'}(X, V')$ be the set of vertices that are (δ, n') heavy in the subgraph induced by V'.

In other words, a vertex is (δ, n')-heavy if it is close to $\delta^2 n'$ other vertices in the given embedding. The next structural property of graphs is what will separate semirandom instances from worst-case instances.

Definition 4.2 (Geometric expansion)**.** A graph $G(V,E)$ on n vertices satisfies the geometric expansion property at scale (δ, n', α) if, for every feasible solution X to Equation (2) on input G and every subset $V' \subseteq V$ such that $H_{\delta,n'}(X,V') = \emptyset$, it holds

$$\left|\left\{ij \in E \cap (V' \times V') \;\middle|\; \|v_i - v_j\|^2 \leqslant \delta\right\}\right| \leqslant 10 \cdot \delta^2 \cdot \alpha\,.$$

That is, a graph is geometrically expanding if the uniform distribution over the edges of non-heavy vertices satisfies a one-sided Chebyshev's inequality. For simplicity, we say that a graph G is geometrically expanding up to scale $(100^{-z}, n, \alpha)$ if it is geometrically expanding at scale $(100^{-i}, n, \alpha)$ for all $1 \leqslant i \leqslant z$. We remark that Definition 4.2 is equivalent to the geometric expansion property defined in (Makarychev et al., 2012).

We are now ready to present the main theorem of the section.

Theorem 4.3 (Main theorem)**.** *There exists a randomized algorithm that on input $a, \alpha > \Omega(1)\,, \kappa\,, \delta > 1/\log n$ and a graph G such that:*

1. *there exists an a-balanced partition (A,B) with $|E(A,B)| \leqslant \alpha\,$,*
2. *$G(V, E(A,B))$ is a geometric expander up to scale $(100\delta, n, \alpha)$,*

returns an $\Omega(a)$-balanced partition (S,T) with cut

$$|E(S,T)| \leqslant O(\alpha)(1 + \delta \cdot \kappa \cdot \sqrt{\log n})$$

with probability $1 - o(1)$. Moreover, the algorithm runs in time $\tilde{O}\left(|V(G)|^{1+O(1/\kappa^2)+o(1)} + |E(G)|^{1+O(1/\kappa^2)+o(1)}\right)$.

Theorem 1.2 essentially follows from Theorem 4.3 observing that graphs generated through Model 1.1 are good geometric expanders. To show this first observe that random bipartite graphs are good geometric expanders.

Theorem 4.4 (Geometric expansion of random graphs, (Makarychev et al., 2012))**.** *Let $t > 0$. Let G be a graph over n vertices generated through the first two steps (i), (ii) of Model 1.1 with parameters $a, \eta > 0$. Then, with probability $1 - n^{-\Omega(1)}$, G is geometrically expanding up to scale $\left(100^{-t}, n, \Theta(n^2 \cdot \eta + 100^t \cdot n \cdot t^2)\right)$.*

Second, observe that geometric expansion in bipartite graphs is a property that is to some extent robust to changes monotone with respect to the bipartition.

Fact 4.5 (Robustness of geometric expansion, (Makarychev et al., 2012)). Let G be a graph over n vertices generated through the first two steps (i), (ii) of Model 1.1 and let G° be a graph obtained after applying steps (iii), (iv). If G is geometrically expanding up to scale (δ, n, τ), then so is $G^\circ(V, E(A,B))$.

This statement above implies that for $\eta \geqslant \Omega\left(\frac{(\log n)^2 \cdot (\log\log n)^2}{n}\right)$, with high probability $G^\circ(V, E(A,B))$ is a good geometric expander. Now Theorem 1.2 immediately follows combining Theorem 4.3 Theorem 4.4 and Fact 4.5.

4.1. The algorithm

We present here the algorithm behind Theorem 4.3. Since we will work using the matrix multiplicative framework (see Appendix A for the necessary definitions), our main challenge is that of designing an appropriate oracle. For simplicity, we split ORACLE in three parts, the first two are due to (Arora & Kale, 2007; Sherman, 2009) the third part is our crucial addition and the main technical contribution of this work. Recall we denote by α the minimum objective of the program at hand.

Lemma 4.6 ((Arora & Kale, 2007))**.** *Let $a > \Omega(1)$. There exists a $\tilde{O}(\alpha/n)$-bounded, $\Theta(\log n)^2$-robust, $\Theta(1)$-separation oracle that, given a candidate solution to Equation (2) with input graph G on n vertices and a-balanced cut of value at most α, outputs **no** if one of the following conditions are violated. Let $W := \left\{i \in [n] \mid \|v_i\|^2 > 2\right\} \subseteq [n]$ and $S := [n] \setminus W$.*

- *Flatness: $|W| < \frac{n}{(\log n)^{100}}$.*
- *Balance: $\sum_{i,j \in S} \|v_i - v_j\|^2 \geqslant 2an$.*

Moreover, the oracle is $\mathcal{T}$-lean for some $\mathcal{T} \leqslant O\left((|V(G)| + |E(G)|\right)$

We omit the proof of Lemma 4.6 as it can be found in (Arora & Kale, 2007). If both the flatness and the balance condition are satisfied, then we apply the following oracle, due to (Sherman, 2009).

Lemma 4.7 ((Sherman, 2009))**.** *Let $\kappa, a > 0\,, 0 < \delta < 1/200, a > \Omega(1)$. There exists a $\tilde{O}(\alpha/n)$-bounded, $O(\log n)^2$-robust, $\Theta(1)$-separation oracle that, given a candidate solution to Equation (3) with input graph G on n vertices and a-balanced cut of value at most α, outputs **yes** only if it finds an $\Omega(a)$-balanced partition (P,P') of $V(G)$ satisfying*

$$\sum_{i \in P\,, j \in P'\,, ij \in E(G)} \|v_i - v_j\|^2 \leqslant O(\alpha)$$

$$|E(P,P')| \leqslant O(\alpha \cdot \kappa) \cdot \sqrt{\log n}\,.$$

Moreover, the oracle is $\mathcal{T}$-lean for some $\mathcal{T} \leqslant \tilde{O}\left(|V(G)|^{1+O(1/\kappa^2)+o(1)} + |E(G)|^{1+O(1/\kappa^2)+o(1)}\right)$.

The proof of Lemma 4.7 can be found in (Sherman, 2009). The improvement on the time complexity follows by Theorem A.1. The next result is the *crucial* addition we need

to the oracle of (Arora & Kale, 2007; Sherman, 2009). We prove it in Appendix C.1.

Lemma 4.8. *Let $0 < \ell \leqslant 1\,, \alpha, a > 0$ and $0 < \delta \leqslant 1/200$. Let G be a graph on $\ell \cdot n$ vertices such that*

- *it has a a-balanced partition (A, B) with $|E(A,B)| \leqslant \alpha\,$,*
- *$G(V, E(A,B))$ is geometrically expanding up to scale (δ, n, α).*

*There exists a $\tilde{O}(\alpha/\ell \cdot n)$-bounded, $O(\log n)^{100}$-robust, $\Theta(1/\log n)$-separation oracle that, given a candidate solution to Equation (3) with input graph G, either outputs **no**, or outputs a set of edges $E^* \subseteq E(G)$ of cardinality $O(\alpha/\delta)$ and partition (P_1, P_2, V') of $V(G)$ such that*

1. $|E(P_1, P_2, V') \setminus E^*| \leqslant O\left(\frac{\alpha}{\delta}\left(1 + \frac{\ell}{\delta}\right)\right)\,.$
2. $||P_1| - |P_2|| \leqslant a \cdot n/2\,.$
3. $\forall ij \in E(G) \setminus E^*$ *with* $i, j \in V'$ *it holds* $\|v_i - v_j\|^2 \leqslant \delta\,.$
4. $H_{\delta,n}(X, V') = \emptyset\,.$

Moreover the oracle is $\mathcal{T}$-lean for some $\mathcal{T} \leqslant \tilde{O}\left(|V(G)| + |E(G)|\right)$ and $1 - O(\log n)^{-50}$-reliable.

Before presenting the algorithm that uses the oracle above, let's briefly discuss its meaning. Notice the *heavy vertices* condition (4). This ensures that in the subgraph $G(V', E \setminus E^*)$ any feasible embedding has weight at most $10\alpha \cdot \delta^2$ on the edges in $(E(A,B) \setminus E^*) \cap (V' \times V')$. In other words, after paying the edges in the cut of the partition (P_1, P_2, V'), geometric expansion of the underlying graph guarantees that the minimum objective value of Equation (2) now decrease by a $10\delta^2$ factor.

Next we present the algorithm behind Theorem 4.3 and prove its correctness. We denote by ORACLE a combination of the oracles in Lemma 4.6, Lemma 4.7 and Lemma 4.8 obtained applying them sequentially (in this order).

Proof of Theorem 4.3. We set T such that $\delta^{(T)} = 100\delta$ where $\delta^{(0)} = 1/200$. By construction of ORACLE and Corollary A.10 the algorithm runs in time $\tilde{O}\left(|V(G)|^{1+O(1/\kappa^2)+o(1)} + |E(G)|^{1+O(1/\kappa^2)+o(1)}\right)$.

Now consider a fixed iteration i, we assume $\alpha^{(0)} = \alpha\,, \ell^{(0)} = 1$. Let $\alpha^{(i)}$ be the cost of the minimum feasible solution on the remaining graph $G^{(i)}$ on $\ell^{(i)} \cdot n$ vertices. Let $E^{(i)}$ be the set of edges removed at iteration i and $(P_1^{(i)}, P_2^{(i)}, V'^{(i)})$ the partition at iteration i. Notice that if

Algorithm 1 Fast and robust algorithm for balanced cut

Input: A graph G with minimum a-balanced cut of value at most $\alpha, T\,, \kappa, d, \delta > 0$.

Set $\delta^{(0)} = 1/200$.
for $i = 1$ **to** T **do**
  Let $G^{(i)}$ be the current remaining graph with optimal cut value $\alpha^{(i)}$ and $\left|V(G^{(i)})\right| =: n^{(i)}$.
  Run the approximate matrix multiplicative weights algorithm (Algorithm 3) for program 3 using ORACLE (with parameter $\delta^{(i)}$).
  Let $W^* \in \mathbb{R}^{d,\times n}$ be the returned embedding, $E^{(i)}$ the set of edges found and $(P^{(i)}, P'^{(i)})$, $(P_1^{(i)}, P_2^{(i)}, V'^{(i)})$ the partitions found by ORACLE .
  Remove the edges in $E^{(i)}$.
  if $\left|E(P^{(i)}, P'^{(i)})\right| \leqslant O(\alpha \cdot (1 + \delta \cdot \kappa \cdot \sqrt{\log n}))$ **then**
    Exit the loop.
  else
    Remove vertices in $P_1^{(i)}$ and in $P_2^{(i)}$. Set $\delta^{(i+1)}$ to $\delta^{(i)}/100$.
  end if
end for
Let (P, P') be the bipartition found by ORACLE in its last iteration.
Arbitrarily assign sets $P_1^{(1)}, P_2^{(1)}, \ldots, P_1^{(i)}, P_2^{(i)}$ removed in previous iterations to P or P', keeping the two sides a-balanced.
Return the resulting bipartition.

at some point $\alpha^{(i)} \leqslant O(\alpha \cdot (1 + \delta \cdot \kappa \cdot \sqrt{\log n}))$ then the algorithm breaks the cycle and returns a balanced partition.

So we may assume that at the current iteration i, $\alpha^{(i)} \geqslant \omega(\alpha \cdot ((1 + \delta \cdot \kappa \cdot \sqrt{\log n}))$. Now the result follows by showing that, at each step, it holds

$$\alpha^{(i)} \leqslant 10 \cdot \alpha^{(i-1)} \cdot \left(\delta^{(i-1)}\right)^2\,. \tag{4}$$

Indeed suppose the claim holds. By construction all the edges in the final cut are in

$$\left(\bigcup_{i<T} E^{(i)}\right) \cup \left(\bigcup_{i<T} E(P_1^{(i)}, P_2^{(i)}, V'^{(i)}) \setminus E^{(i)}\right) \cup E(P^{(T)}, P'^{(T)})\,.$$

By Equation (4) we can bound the first term as $|\bigcup_{i \leqslant T} E^{(i)}| \leqslant O(\alpha)$. For the second term:

$$\left|\bigcup_{i \leqslant T} E(P_1^{(i)}, P_2^{(i)}, V'^{(i)})\right| \leqslant \sum_{i \leqslant T} O\left(\frac{\alpha^{(i)}}{\delta^{(i)}}\left(1 + \frac{\ell^{(i)}}{\delta^{(i)}}\right)\right)$$

$$
\begin{aligned}
&\leqslant O\left(\sum_{i\leqslant T}\frac{\alpha^{(i)}}{\delta^{(i)}}+\frac{\alpha^{(i)}\cdot\ell^{(i)}}{\left(\delta^{(i)}\right)^2}\right)\\
&\leqslant O\left(\sum_{i\leqslant T}\alpha^{(i)}\left(\delta^{(i)}+\ell^{(i+1)}\right)\right)\\
&\leqslant O\left(\sum_{i\leqslant T}\alpha^{(i)}\cdot\left(\delta^{(i)}+\ell^{(i)}\right)\right)\\
&\leqslant \alpha\cdot O\left(\sum_{i\leqslant T}\delta^{(i)}+\ell^{(i)}\right)\\
&\leqslant O(\alpha)\,,
\end{aligned}
$$

where in the second step we used the inequalities

$$
\begin{aligned}
\frac{\alpha^{(i)}}{\delta^{(i)}}&\leqslant 10^3\cdot\alpha^{(i-1)}\cdot\delta^{(i-1)}\,,\\
\frac{\alpha^{(i)}\cdot\ell^{(i)}}{\left(\delta^{(i)}\right)^2}&\leqslant 10^5\cdot\alpha^{(i-1)}\cdot\ell^{(i)}\,,
\end{aligned}
$$

both following from Equation (4). For the third term we have $\alpha^{(T)}\leqslant O\left(\alpha\cdot\delta\right)$ by construction. Thus by Lemma 4.7 we get $\left|E(P^{(T)},P'^{(T)})\right|\leqslant O\left(\alpha\cdot\kappa\cdot\delta\sqrt{\log n}\right)$.

It remains to prove Equation (4). At each iteration i, the set $V(G^{(i)})$ does not contain edges of length more than $\delta^{(i)}$ in the embedding as well as $(\delta^{(i)},n)$ heavy vertices. Thus by Definition 4.2, the set $V(G^{(i)})$ has a $\Omega(a)$-balanced cut of value $O(\alpha^{(i)})$. Then Equation (4) follows as desired. □

Acknowledgments

Tommaso d'Orsi is partially supported by the project MUR FARE2020 PAReCoDi.

Impact statement

This paper presents work whose goal is to advance the field of theoretical Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

References

Abbe, E. Community detection and stochastic block models: recent developments. *The Journal of Machine Learning Research*, 18(1):6446–6531, 2017.

Agarwal, A., Charikar, M., Makarychev, K., and Makarychev, Y. $o(\sqrt{\log n})$ approximation algorithms for min uncut, min 2cnf deletion, and directed cut problems. In *Proceedings of the thirty-seventh annual ACM symposium on Theory of computing*, pp. 573–581, 2005.

Angelidakis, H., Makarychev, K., and Makarychev, Y. Algorithms for stable and perturbation-resilient problems. In Hatami, H., McKenzie, P., and King, V. (eds.), *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017*, pp. 438–451. ACM, 2017. doi: 10.1145/3055399.3055487. URL https://doi.org/10.1145/3055399.3055487.

Arora, S. and Kale, S. A combinatorial, primal-dual approach to semidefinite programs. In *Proceedings of the 39th Annual ACM Symposium on Theory of Computing, San Diego, California, USA, June 11-13, 2007*, pp. 227–236, 2007. doi: 10.1145/1250790.1250823. URL https://doi.org/10.1145/1250790.1250823.

Arora, S., Rao, S., and Vazirani, U. V. Expander flows, geometric embeddings and graph partitioning. In *Proceedings of the 36th Annual ACM Symposium on Theory of Computing, Chicago, IL, USA, June 13-16, 2004*, pp. 222–231, 2004. doi: 10.1145/1007352.1007355. URL https://doi.org/10.1145/1007352.1007355.

Bakshi, A., Diakonikolas, I., Jia, H., Kane, D. M., Kothari, P. K., and Vempala, S. S. Robustly learning mixtures of k arbitrary gaussians. In *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing*, pp. 1234–1247, 2022.

Bilu, Y. and Linial, N. Are stable instances easy? *Comb. Probab. Comput.*, 21(5):643–660, 2012a. doi: 10.1017/S0963548312000193. URL https://doi.org/10.1017/S0963548312000193.

Bilu, Y. and Linial, N. Are stable instances easy? *Combinatorics, Probability and Computing*, 21(5):643–660, 2012b.

Boppana, R. B. Eigenvalues and graph bisection: An average-case analysis. In *28th Annual Symposium on Foundations of Computer Science (sfcs 1987)*, pp. 280–285. IEEE, 1987.

Bui, T. N., Chaudhuri, S., Leighton, F. T., and Sipser, M. Graph bisection algorithms with good average case behavior. *Combinatorica*, 7(2):171–191, 1987.

Chen, L., Kyng, R., Liu, Y. P., Peng, R., Gutenberg, M. P., and Sachdeva, S. Maximum flow and minimum-cost flow in almost-linear time. In *2022 IEEE 63rd Annual Symposium on Foundations of Computer Science (FOCS)*, pp. 612–623. IEEE, 2022.

Chierichetti, F., Panconesi, A., Re, G., and Trevisan, L. Spectral robustness for correlation clustering reconstruction in semi-adversarial models. In Camps-Valls, G.,

Ruiz, F. J. R., and Valera, I. (eds.), *International Conference on Artificial Intelligence and Statistics, AISTATS 2022, 28-30 March 2022, Virtual Event*, volume 151 of *Proceedings of Machine Learning Research*, pp. 10852–10880. PMLR, 2022. URL https://proceedings.mlr.press/v151/chierichetti22a.html.

Cohen-Addad, V., Kanade, V., Mallmann-Trenn, F., and Mathieu, C. Hierarchical clustering: Objective functions and algorithms. *Journal of the ACM (JACM)*, 66(4):1–42, 2019.

Dasgupta, S. A cost function for similarity-based hierarchical clustering. In Wichs, D. and Mansour, Y. (eds.), *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016*, pp. 118–127. ACM, 2016. doi: 10.1145/2897518.2897527. URL https://doi.org/10.1145/2897518.2897527.

Decelle, A., Krzakala, F., Moore, C., and Zdeborová, L. Asymptotic analysis of the stochastic block model for modular networks and its algorithmic applications. *Physical Review E*, 84(6):066106, 2011.

Diakonikolas, I. and Kane, D. M. Recent advances in algorithmic high-dimensional robust statistics. *arXiv preprint arXiv:1911.05911*, 2019.

Dimitriou, T. and Impagliazzo, R. Go with the winners for graph bisection. In *SODA*, volume 98, pp. 510–520, 1998.

Ding, J., d'Orsi, T., Nasser, R., and Steurer, D. Robust recovery for stochastic block models. In *2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS)*, pp. 387–394. IEEE, 2022.

d'Orsi, T., Kothari, P. K., Novikov, G., and Steurer, D. Sparse pca: algorithms, adversarial perturbations and certificates. In *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*, pp. 553–564. IEEE, 2020.

Dyer, M. E. and Frieze, A. M. Fast solution of some random np-hard problems. In *27th Annual Symposium on Foundations of Computer Science (sfcs 1986)*, pp. 331–336. IEEE Computer Society, 1986.

Fei, Y. and Chen, Y. Achieving the bayes error rate in stochastic block model by sdp, robustly. In Beygelzimer, A. and Hsu, D. (eds.), *Conference on Learning Theory, COLT 2019, 25-28 June 2019, Phoenix, AZ, USA*, volume 99 of *Proceedings of Machine Learning Research*, pp. 1235–1269. PMLR, 2019. URL http://proceedings.mlr.press/v99/fei19a.html.

Feige, U. and Kilian, J. Heuristics for semirandom graph problems. *Journal of Computer and System Sciences*, 63 (4):639–671, 2001.

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

Goemans, M. X. and Williamson, D. P. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM (JACM)*, 42(6):1115–1145, 1995.

Guruswami, V., Kothari, P. K., and Manohar, P. Algorithms and certificates for boolean csp refutation: smoothed is no harder than random. In *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing*, pp. 678–689, 2022.

Hopkins, S. B. and Steurer, D. Efficient bayesian estimation from few samples: community detection and related problems. In *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, pp. 379–390. IEEE, 2017.

Hua, Y., Ding, J., d'Orsi, T., and Steurer, D. Reaching kesten-stigum threshold in the stochastic block model under node corruptions. In *The Thirty Sixth Annual Conference on Learning Theory*, pp. 4044–4071. PMLR, 2023.

Jerrum, M. and Sorkin, G. B. *Simulated annealing for graph bisection*. IEEE, 1993.

Johnson, W. B. Extensions of lipschitz mappings into a hilbert space. *Contemp. Math.*, 26:189–206, 1984.

Khot, S., Kindler, G., Mossel, E., and O'Donnell, R. Optimal inapproximability results for max-cut and other 2-variable csps? *SIAM Journal on Computing*, 37(1): 319–357, 2007.

Khot, S. A. and Vishnoi, N. K. The unique games conjecture, integrality gap for cut problems and embeddability of negative-type metrics into ℓ-1. *Journal of the ACM (JACM)*, 62(1):1–39, 2015.

Leighton, F. T. and Rao, S. An approximate max-flow min-cut theorem for uniform multicommodity flow problems with applications to approximation algorithms. In *29th Annual Symposium on Foundations of Computer Science, White Plains, New York, USA, 24-26 October 1988*, pp. 422–431, 1988. doi: 10.1109/SFCS.1988.21958. URL https://doi.org/10.1109/SFCS.1988.21958.

Liu, A. and Moitra, A. Minimax rates for robust community detection. In *2022 IEEE 63rd Annual Symposium on Foundations of Computer Science (FOCS)*, pp. 823–831. IEEE, 2022.

Makarychev, K., Makarychev, Y., and Vijayaraghavan, A. Approximation algorithms for semi-random partitioning problems. In *Proceedings of the 44th Symposium on Theory of Computing Conference, STOC 2012, New York, NY, USA, May 19 - 22, 2012*, pp. 367–384, 2012. doi: 10.1145/2213977.2214013. URL https://doi.org/10.1145/2213977.2214013.

Makarychev, K., Makarychev, Y., and Vijayaraghavan, A. Constant factor approximation for balanced cut in the PIE model. In Shmoys, D. B. (ed.), *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*, pp. 41–49. ACM, 2014. doi: 10.1145/2591796.2591841. URL https://doi.org/10.1145/2591796.2591841.

Massoulié, L. Community detection thresholds and the weak ramanujan property. In *Proceedings of the forty-sixth annual ACM symposium on Theory of computing*, pp. 694–703, 2014.

McSherry, F. Spectral partitioning of random graphs. In *Proceedings 42nd IEEE Symposium on Foundations of Computer Science*, pp. 529–537. IEEE, 2001.

Moitra, A., Perry, W., and Wein, A. S. How robust are reconstruction thresholds for community detection? In *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*, pp. 828–841, 2016.

Montanari, A. and Sen, S. Semidefinite programs on sparse random graphs and their application to community detection. In *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*, pp. 814–827, 2016.

Mossel, E., Neeman, J., and Sly, A. Reconstruction and estimation in the planted partition model. *Probability Theory and Related Fields*, 162(3):431–461, 2015.

Mossel, E., Neeman, J., and Sly, A. A proof of the block model threshold conjecture. *Combinatorica*, 38(3):665–708, 2018.

Peng, P. Robust clustering oracle and local reconstructor of cluster structure of graphs. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 2953–2972. SIAM, 2020.

Räcke, H. Optimal hierarchical decompositions for congestion minimization in networks. In *Proceedings of the fortieth annual ACM symposium on Theory of computing*, pp. 255–264, 2008.

Raghavendra, P. Optimal algorithms and inapproximability results for every csp? In *Proceedings of the fortieth annual ACM symposium on Theory of computing*, pp. 245–254, 2008.

Raghavendra, P. and Steurer, D. Graph expansion and the unique games conjecture. In *Proceedings of the forty-second ACM symposium on Theory of computing*, pp. 755–764, 2010.

Raghavendra, P., Steurer, D., and Tulsiani, M. Reductions between expansion problems. In *2012 IEEE 27th Conference on Computational Complexity*, pp. 64–73. IEEE, 2012.

Sherman, J. Breaking the multicommodity flow barrier for o(vlog n)-approximations to sparsest cut. In *50th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2009, October 25-27, 2009, Atlanta, Georgia, USA*, pp. 363–372, 2009. doi: 10.1109/FOCS.2009.66. URL https://doi.org/10.1109/FOCS.2009.66.

Steurer, D. Fast SDP algorithms for constraint satisfaction problems. In *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2010, Austin, Texas, USA, January 17-19, 2010*, pp. 684–697, 2010. doi: 10.1137/1.9781611973075.56. URL https://doi.org/10.1137/1.9781611973075.56.

A. Background

We introduce here background notion used throughout the paper.

Maximum flow Let $G(V, E)$ be a graph. For a flow which assigns value f_p to path p define f_e to be the flow on edge $e \in E(G)$, i.e. $f_e := \sum_{p \ni e} f_p$. Define f_{ij} to be the total flow between nodes $i\,, j$, i.e. $f_{ij} = \sum_{p \in P_{ij}} f_p$, where P_{ij} is the set of paths from i to j. Similarly, define f_i to be flow from node i. That is, $f_i = \sum_{j \in [n]} f_{ij}$. A valid d-regular flow is one that satisfies the capacity constraints: $\forall e \in E \,:\, f_e \leqslant 1$ and $\forall i \in V \,:\, f_i \leqslant d$. For a partition (A, B) of G, the maximum d-regular flow between A and B is the maximum d-regular flow between vertices s and t in the graph obtained from G as follows: (1) connect all vertices in A to a new vertex s by edges of capacity d, (2) connect all vertices in B to a new vertex t by edges of capacity d.

Through the paper we always assume the capacities d to be integral and bounded by $O(\mathrm{poly}(n))$. We assume the algorithm used to compute the maximum flow is the near linear time algorithm in (Chen et al., 2022), captured by the result below:

Theorem A.1 (Maximum flow in almost linear time (Chen et al., 2022))**.** *Let G be a graph on n vertices and let d be integral of value at most $O(\mathrm{poly}(n))$. There exists an algorithm computing the maximum d-regular flow between two vertices in time at most $O(|E(G)|^{1+o(1)})$.*

A.1. The matrix multiplicative weights method for SDPs

We recall here how the matrix multiplicative method can be used to approximately solve semidefinite programs. (Arora & Kale, 2007; Steurer, 2010). As most of the notions presented here already appeared in (Arora & Kale, 2007; Steurer, 2010), we encourage the knowledgeable reader to skip this section, move directly to Section 4 and come back when needed. We focus on minimization problems although the same framework applies to maximization problems.

A primal semidefinite program over n^2 variables (i.e. the n-by-n matrix variable X) and m constraints can be written in its canonical form as

$$\left\{\begin{array}{ll} \min & \langle L, X\rangle \\ \forall j \in [m]\,, & \langle A_j, X\rangle \geqslant b_j \\ & X \succeq \mathbf{0} \end{array}\right\} \tag{5}$$

Here $A_1, \ldots, A_m, L$ are symmetric matrices. We denote the feasible set of solutions by $\mathcal{X}$ and the optimal objective value by α. For simplicity we assume that $A_1 = -\mathrm{Id}_n$ and $b_1 = -r$. This serves to bound the trace of the solution so that $\mathcal{X} \subseteq \Delta_n(r)$. The associated dual, with variables $y_1, \ldots, y_m$, is the following program

$$\left\{\begin{array}{ll} \max & \langle b, y\rangle \\ & \sum_{j \in [m]} A_j y_j \preceq L \\ & y \geqslant \mathbf{0} \end{array}\right\} \tag{6}$$

where b is the m-dimensional vector with entries $b_1, \ldots, b_m$.

For a convex set $\mathcal{X}^* \subseteq \Delta_n(r)$ (think of $\mathcal{X}^*$ as the set of feasible solution to a program of the form Equation (5) with objective value close to the optimum) a γ-*separation* ORACLE is an algorithm that , given a candidate matrix X, outputs one of the following:

- **yes**: the ORACLE determines X is "close" (the precise notion of closeness is problem dependent) to $\mathcal{X}^*$.
- **no**: the ORACLE finds a hyperplane that separates X from $\mathcal{X}^*$ by a γ-margin. That is, it outputs a symmetric matrix M such that for all $X' \in \mathcal{X}^*$ we have $\langle M, X'\rangle \geqslant 0$ while $\langle M, X\rangle < -\gamma\alpha$.

A γ-separation ORACLE is said to be ζ-bounded if $\|M\| \leqslant \zeta$ for any hyperplane M found by the ORACLE . The boundedness of the ORACLE will be relevant for the running time of our algorithms. It is important to notice that the parameters ζ, γ are not independent, in particular one may increase γ by scaling up the corresponding matrix M. We keep them distinct for convenience .

Concretely, given a program of the form Equation (5) and a candidate solution X, we will consider ORACLE algorithms that, in the **no** case, find a pair (y, F) where F is a matrix in $\mathcal{S}_n$ satisfying $F \preceq L$ and y is a candidate solution[10] for the dual program Equation (6) such that $y \in \{y \mid \langle b, y\rangle \geqslant \alpha\,, y \geqslant 0\}$ and

$$\langle \sum_{j\in[m]} A_j y_j - F, X\rangle \leqslant -\gamma \cdot \alpha\,.$$

It is easy to see that this is indeed a separating hyperplane as for any feasible solution X' with objective value less than $\alpha(1+\gamma)$

$$\langle \sum_{j\in[m]} A_j y_j - F, X'\rangle \geqslant \sum_{j\in[m]} b_j y_j - \langle L, X'\rangle > \alpha - (1+\gamma)\alpha = -\gamma\cdot\alpha$$

,.

We will use our oracle algorithms in the following framework.

Algorithm 2 Matrix multiplicative weights algorithm for SDPs

Input: A program of the form Equation (5) with optimal value α, a ζ-bounded γ-separation ORACLE , parameters T, ε, r.

Set $X^{(1)} = \frac{r}{n}\mathrm{Id}_n$.
for $t = 1$ **to** T **do**
  Run the ORACLE with candidate solution $X^{(t)}$.
  if the ORACLE outputs **yes then**
    Return $X^{(t)}$.
  else
    Let $(y^{(t)}, F)$ be the pair generated by ORACLE . Set $Y^{(t)} = \left(\sum_{j\in[m]} A_j y_j^{(t)} - F + \zeta \mathrm{Id}_n\right)/2\zeta$.
  end if
  Compute $X^{(t+1)} = r \cdot \exp\left(\varepsilon \sum_{t'\leqslant t} Y^{(t')}\right) / \operatorname{Tr}\exp\left(\varepsilon \sum_{t'\leqslant t} Y^{(t')}\right)$
end for

The choice of the iterative updates in step 4 is based on the matrix multiplicative weights method. In particular, this allows one to obtain the following crucial statement.

Theorem A.2 ((Arora & Kale, 2007))**.** *Consider Algorithm 2. Let $\varepsilon \leqslant \gamma\alpha/(2\zeta\cdot r)$ and $T \geqslant 2\varepsilon^{-2}\log n$. If there exists a feasible solution with value at most $\alpha(1+\gamma)$, then* ORACLE *will output* ***yes*** *within T iterations.*

A.1.1. Approximate matrix exponentiation, robust and reliable oracles

There are two issues with Theorem A.2 if one aims for near linear running time: first, already writing down $X^{(t)}$ requires time quadratic in n; second, algorithms known to compute the matrix exponentiation are slow . One can circumvent these obstacles computing the exponentiation only approximately while also keeping only an approximate representation of $X^{(t)}$. To formalize this we introduce additional notation. For a positive semidefinite n-by-n matrix M, we let $P_{\leqslant p}(M)$ be the degree-p approximation of the matrix exponential $\exp(X)$:

$$P_{\leqslant p}(M) := \sum_{i\leqslant p} \frac{1}{i!} M^i\,.$$

Recall that for a matrix M, the Gram decomposition of the exponential $\exp(M)$ is $\exp(M) = \exp(\frac{1}{2}M)^{\mathsf{T}}\exp(\frac{1}{2}M)$ thus we may see $P_{\leqslant p}(\frac{\varepsilon}{2}\sum_{t'\leqslant t} Y^{(t')})$ as a a matrix having as columns low-degree approximations of the Gram vectors of $\exp(M)$. One can then embed these vectors in a low dimensional space, without distorting their pair-wise distance by projecting them onto a random d-dimensional subspace:

[10] Not necessarily feasible.

Lemma A.3 ((Johnson, 1984)). *Let Φ be a d-by-n Gaussian matrix, with each entry independently chosen from $N(0,1/d)$. Then, for every vector $u \in \mathbb{R}^n$ and every $\varepsilon \in (0,1)$*

$$\mathbb{P}\left(\|\Phi u\| = (1 \pm \varepsilon)\|u\|\right) \geqslant 1 - e^{-\Omega(\varepsilon^2 d)}.$$

We will follow this strategy to speed up Algorithm 2.

Algorithm 3 Approximate matrix multiplicative weights algorithm for SDPs

Input: A program of the form Equation (5) with optimal value α, a ζ-bounded γ-separation ORACLE , parameters T, ε, r, d, p, a d-by-n random matrix Φ with i.i.d entries from $N(0,1/d)$.

Set $W^{(1)} = \frac{r}{n}(\Phi \mathrm{Id}_n)/\operatorname{Tr}(\Phi \mathrm{Id}_n)$.
for $t = 1$ **to** T **do**
  Run the ORACLE with candidate solution $W^{(t)}$.
  if the ORACLE outputs **yes then**
    Return $W^{(t)}$.
  else
    Let $(y^{(t)}, F)$ be the pair generated by ORACLE . Set $Y^{(t)} = \left(\sum_{j\in[m]} A_j y_j^{(t)} - F + \zeta \mathrm{Id}_n\right)/2\zeta$.
  end if
  Sample a d-by-n random matrix Φ with i.i.d entries from $N(0,1/d)$.
  Compute $W^{(t)} = r \cdot \Phi P_{\leqslant p}(\frac{\varepsilon}{2}\sum_{t'\leqslant t} Y^{(t')})/\operatorname{Tr}\left(\Phi P_{\leqslant p}(\frac{\varepsilon}{2}\sum_{t'\leqslant t} Y^{(t')})\right)$.
end for

Observe that $P_{\leqslant p}(\frac{\varepsilon}{2}\sum_{t'\leqslant t} Y^{(t')})$ corresponds to a low degree approximation of the Gram vectors of the matrix $X^{(t)}$ in step 4 of Algorithm 2. We then compute $W^{(t)}$ by embedding these vectors in a random d-dimensional space.

The statement below shows that in many cases we can compute such matrices $W^{(t)}$ very efficiently.

Lemma A.4 ((Steurer, 2010)). *Suppose we can perform matrix-vector multiplication with the matrices $Y^{(t)}$ in time $\mathcal{T}$. Then, for every t, we can compute $W^{(t)}$ in time $O(t \cdot p \cdot d \cdot \mathcal{T})$.*

A priori it is not clear whether Algorithm 3 can provide the same guarantees of Algorithm 2. However, the next result show this is the case under reasonable circumstances.

Definition A.5 (d-robust oracle, extension of (Steurer, 2010)). We say that a ζ-bounded γ-separation oracle is d-robust if for every matrix $X \in \Delta(r)$ with $X = W^{\mathsf{T}}W$

$$\mathbb{P}_{\Phi\sim N(0,1/d)^{d\times n}}\left(\text{ORACLE outputs } \textbf{no} \text{ on input } (\Phi W)^{\mathsf{T}}\Phi W \text{ and } \langle Y^{(t)}, X\rangle \geqslant -\frac{3}{4}\gamma\alpha\right) \leqslant \frac{(\gamma\alpha/\zeta r)^2}{(\log n)^{10}}.$$

Lemma A.6 ((Steurer, 2010)). *Consider Algorithm 3. Let $\varepsilon \leqslant \gamma\alpha/(2\zeta \cdot r)$, $T \geqslant 2\varepsilon^{-2}\log n$ and $p \geqslant 10\varepsilon^{-1}\log n$. Suppose we have a d-robust ζ-bounded γ-separation* ORACLE *. If there exists a feasible solution with value at most $\alpha(1+2\gamma)$, then* ORACLE *will output **yes** within T iterations with probability at least $1 - O(\log n)^{-10}$.*

We can combine Lemma A.6, Lemma A.4 and Fact A.8 to obtain a user-friendly statement concerning the running time of Algorithm 3. We introduce two additional definitions.

Definition A.7 ($\mathcal{T}$-lean oracle). We say that a ζ-bounded γ-separation d-robust is $\mathcal{T}$-lean if:

- the oracle compute its outputs in time at most $O(\mathcal{T})$.
- If the oracle outputs **no**, the matrix-vector multiplication between an arbitrary vector and the feedback matrix $\left(\sum_{j\in[m]} A_j y_j - F + \zeta \mathrm{Id}_n\right)/2\zeta$ can be computed in time $O(\mathcal{T})$.

We remark that one can upper bound the time needed for matrix-vector multiplication by the number of non-zero entries in the matrix of interest.

Fact A.8. Let $M \in \mathbb{R}^{n\times n}$ be a matrix with m non-zero entries and let $v \in \mathbb{R}^n$. There exists an algorithm that computes Mv in time $O(m+n)$.

The next definition formalizes the idea of oracles that may find a separating hyperplane only with certain probability.

Definition A.9 (q-reliable)**.** We say that a ζ-bounded, γ-separation, d-robust, $\mathcal{T}$-lean oracle is q-reliable if the probability (over random bits) that it outputs **no** for any feasible solution with objective value at most $(1+2\gamma)\alpha$ is at most $1-q$.

For oracles that are 1-reliable we omit mentioning their reliability. We are ready to present a user-friendly running time statement, which we will use as a black box.

Corollary A.10 (Running time of Algorithm 3)**.** *Let* ORACLE *be a ζ-bounded, γ-separation, d-robust, $\mathcal{T}$-lean q-reliable oracle. Then, for $\varepsilon \leqslant \gamma\alpha/(2\zeta\cdot r)$, $T \geqslant 2\varepsilon^{-2}\log n$ and $p \geqslant 10\varepsilon^{-1}\log n$, with probability at least $1-O(\log n)^{-10}-(1-q)T$ over random bits, Algorithm 3 terminates in time $O\left(T^2\cdot\mathcal{T}\cdot d\cdot p\right)$.*

Proof. The Corollary follows immediately from Lemma A.4, Lemma A.6, Definition A.7 and Definition A.9. □

B. Proof of Theorem 4.3

We show here correctness of Algorithm 1, thus obtaining Theorem 4.3.

C. The heavy vertices removal oracle

We prove here Lemma 4.8. In Appendix C.1 we introduce a procedure that the oracle uses to find either the partition or a separating hyperplane. Then in Appendix C.2 we prove the Lemma. Throughout the section we consider the following parameters range:

$$n, \alpha > 0\,, \Omega(1) \leqslant a \leqslant 1\,, 0 < \ell \leqslant 1\,, \Omega(1\log n) \leqslant \delta \leqslant 1/200\,. \tag{7}$$

C.1. The fast heavy vertices removal procedure

We introduce the main procedure used by ORACLE . The central tool of the section is the following statement.

Lemma C.1. *Consider the parameter settings of Equation (7). Let G be a graph on $\ell\cdot n$ vertices with a-balanced cut of value at most α that is geometrically expanding up to scale (δ, n, α).*

Let X be a feasible solution for Equation (2) on input G, with objective value $O(\alpha)$. There exists a randomized procedure (Algorithm 4) that outputs a set of edges $E^* \subseteq E(G)$ of cardinality $O(\alpha/\delta)$ and a partition (P_1, P_2, V') of $V(G)$ satisfying $(2)\,,(3)\,,(4)$ in Lemma 4.8 and such that

$$\mathbb{E}\left[|E(P_1,P_2,V')\setminus E^*|\right] \leqslant C\cdot\frac{\alpha}{\delta}\left(1+\frac{\ell}{\delta}\right)\,,$$

where $C>0$ is a universal constant. Moreover, if the solution is given in the form of $v_1,\ldots,v_n \in \mathbb{R}^{O(\mathrm{polylog}\, n)}$, the procedure runs in time $\tilde{O}(|V(G)|+|E(G)|)$.

The first building block towards a proof of Lemma C.1 is the result below, which introduces a subroutine to identify heavy vertices.

Lemma C.2. *Consider the settings of Lemma C.1. Let $\rho \geqslant 2$. There exists a randomized procedure that outputs with probability at least $1-1/n$ a set of vertices V^* and a mapping $f: V \to V^* \cup \{(*)\}$ such that*

1. *Each vertex i of V^* satisfies $\left|\left\{j\in V \;\middle|\; \|v_i-v_j\|^2 \leqslant \rho\delta\right\}\right| \geqslant 10\delta^2 n$; and*
2. *The set $W := \{i \mid f(i)=(*)\}$ does not contain a vertex i such that $\left|\left\{j\in V \;\middle|\; \|v_i-v_j\|^2 \leqslant \delta\right\}\right| \geqslant 10\delta^2 n$.*

3. $f(i) = j$ *if there exists some* $j \in V^*$ *with* $\|v_i - v_j\|^2 \leqslant \rho\delta$,

4. $f(i) = (*)$ *otherwise.*

Moreover, if the solution is given in the form of $v_1, \ldots, v_n \in \mathbb{R}^{O(\text{polylog}\, n)}$*, the procedure runs in time* $O\left(\frac{1}{\delta^2} \cdot |V(G)| \,\text{polylog}\, n\right)$.

Proof. We propose and analyze the following algorithm:

1. $S \leftarrow$ Sample $100\delta^{-2} \log n$ points uniformly at random.

2. For each point $i \in S$, compute $N(v) = \left\{ j \in W \;\middle|\; \|v_i - v_j\|^2 \leqslant \rho\delta \right\}$.

3. $V^* \leftarrow S \setminus \left\{ i \in S \;\middle|\; |N(v)| < 10\delta^2 n \right\}$

4. For each vertex i, if there exists $j \in V^*$ such that $\|v_i - v_j\|^2 \leqslant \rho\delta$ then $f(i) = j$ otherwise $f(i) = (*)$.

Clearly the above procedure runs in time $O(|V(G)| \cdot |S| \cdot \text{polylog}\, n)$ as desired, where the bulk of the work is done in the second and fourth steps. By the definition of the procedure, the first, third and fourth bullets of the theorem statement are satisfied. We thus need to show that the set $W := \{i \mid f(i) = (*)\}$ does not contain a vertex i such that $\left|\left\{ j \in W \;\middle|\; \|v_i - v_j\|^2 \leqslant \delta \right\}\right| \geqslant 10\delta^2 n$.

A simple coupon collector argument implies that with probability at least $1 - 1/n^2$, for each vertex i, if $\left|\left\{ j \in V(G) \;\middle|\; \|v_i - v_j\|^2 \leqslant \delta \right\}\right| \geqslant 10\delta^2 n$, then $\left\{ j \in V(G) \;\middle|\; \|v_i - v_j\|^2 \leqslant \delta \right\} \cap S \neq \emptyset$. Thus, let j^* be a vertex in $\left\{ j \in V(G) \;\middle|\; \|v_i - v_j\|^2 \leqslant \delta \right\} \cap S \neq \emptyset$. Then, since $\rho \geqslant 2$, we have that $\left|\left\{ j \in V(G) \;\middle|\; \|v_{j^*} - v_j\|^2 \leqslant \rho\delta \right\}\right| \geqslant 10\delta^2 n$ and so $j^* \in V^*$ and $f(i) \neq (*)$ as desired. It remains to take a union bound over the probability of failure for each individual vertex, and we conclude that the overall failure probability is at most $1/n$.

□

We use the procedure in Lemma C.2 as a subroutine of the one presented next, which for feasible embeddings finds a paritition satisfying (2) , (3) , (4) in Lemma 4.8 and (1) in expectation.

Algorithm 4 Fast heavy vertex removal procedure

Input: A graph G on $\ell \cdot n$ vertices, a candidate solution X to Equation (2) on input G, parameters $a, \delta > 0$, $C > 200$.

Remove all edges of length at least δ in the embedding. Let E^* be the set of such edges.
Find the set V^* via the subroutine in Lemma C.2 with $\rho = 2$.
Pick a maximal set U of vertices in V^* at pairwise squared distance at least $10 \cdot C\delta$ in the embedding.
if $|U| \geqslant \frac{a}{C \cdot \delta}$: **then**
  Pick $r \overset{u.a.r.}{\sim} [1, 2]$.
  For each $i \in U$, remove i and all vertices at distance $\leqslant 2r \cdot \delta$ in the embedding. Let U_i be the set of removed vertices via i.
  Repeat the algorithm on the remaining graph.
else
  Run the subroutine Algorithm 5 on the remaining graph and obtain additional sets U_i's.
end if
Distribute evenly the vertices in the U_i's among two sets P_1, P_2 so that if $j, k \in U_i$ then j, k are in the same set. Let $V' = V \setminus (P_1 \cup P_2)$.
Return the partition (P_1, P_2, V') .

Fact C.3. Algorithm 4 runs in time $\tilde{O}\left(|V(G)| + |E(G)|\right)$.

Proof. Step 1 requires $O(E)$ time. The steps 2-4 can be repeated at most $C \cdot \ell/(a \cdot \delta)$ times. Indeed no vertex can be in both U_i and $U_{i'}$ at the same time (even if X does not satisfy the triangle inequality constraints) and since by definition each U_i contains at least $10\delta^2 \cdot n$ vertices, in $C \cdot \ell/(a \cdot \delta)$ iterations we will have removed all vertices form the graph. For each of these iterations, step 2 requires time $\tilde{O}\left(\frac{1}{\delta^2} |V(G)|\right)$ and step 3 requires time $O(|V(G)| \cdot \mathrm{poly}(1/a\delta)$. Step 4 runs in time at most $O(|V(G)|/a\delta)$.

As we show in Fact C.4, Algorithm 5 also runs in time $\tilde{O}(|V(G)| + |E(G)|)$. Step 6 can be done in time $\tilde{O}(|V(G)|)$ after ordering the sets U_i's. Thus the statement follows as $\delta \geqslant 1/\operatorname{poly}\log(n)$, $a \geqslant \Omega(1)$. □

The subroutine of step 5 in Algorithm 4 is presented below.

Algorithm 5 Subroutine of fast heavy vertex removal procedure

Input: A graph G on $\ell \cdot n$ vertices, a candidate solution X to Equation (2) on input G, the list of vertices V^*, parameters $a, \delta > 0$.

Consider the graph $G^*(V^*, \emptyset)$.
for $ij \in E(G)$ **do**
  if $\|v_i - v_j\|^2 \leqslant 200\delta$ and $f(i) \neq (*)$, $f(j) \neq (*)$ **then**
    Connect $f(i)$ to $f(j)$ in G^* (excluding self-loops).
  end if
end for
Pick $r \overset{u.a.r.}{\sim} [1, 2]$ and for each connected component U in G^*, remove all vertices at distance $\leqslant 2r \cdot \delta$ to some vertex $i \in U$. Index the resulting sets by arbitrary representative vertices in each component.
Return the resulting sets U_i's.

Fact C.4. Algorithm 5 runs in time $\tilde{O}(|V(G)| + |E(G)|)$.

Proof. We use the mapping of Lemma C.2. We can then construct the graph G^* in time $O(|E(G)|)$. Moreover, notice that $|E(G^*)| \leqslant |E(G)|$. We can find the connected components in G^* in time $O(|E(G^*)| + |V(G^*)|)$ and partition the vertices in G according to such connected components in time $\tilde{O}(|V(G)|)$. The result follows. □

Next we bound the probability that an edge gets cut in Algorithm 4.

Lemma C.5. *Consider the settings of Lemma C.1. At each iteration of steps 2-4 in Algorithm 4 as well as the one using Algorithm 5 the following holds:*

$$\forall i \text{ s. t. } f_i \neq (*) \quad \mathbb{P}\left(\exists U_k \in U \text{ s. t. } i \in U_k, j \notin U_k\right) \leqslant \frac{\|v_i - v_j\|^2}{2 \cdot \delta} .$$

Proof. Consider first an iteration of steps 1-3 in Algorithm 4. By construction each vertex i can be in at most one set U_k. Since r is chosen uniformly at random in the interval $[1, 2]$ the claim follows. So consider Algorithm 5. Again, by construction each vertex i can be in at most one set U_k so by choice of r the inequality holds. □

Now Lemma C.1 follows as a simple corollary.

Proof of Lemma C.1. By Fact C.3, steps 1-3 in Algorithm 4 are repeated at most $C \cdot \ell/\delta$ times while Algorithm 5 runs only once. By Lemma C.5 we then have $\forall ij \in E(G) \setminus E^*$

$$\mathbb{P}\left(ij \in E(P_1, P_2, V') \setminus E^*\right) \leqslant C^* \frac{\|v_i - v_j\|^2}{\delta} \cdot (1 + C \cdot \ell/\delta) ,$$

for some $C^* > 0$. Then the bound on $\mathbb{E}\left[|E(P_1, P_2, V') \setminus E^*|\right]$ follows by linearity of expectation. By definition of V^*, the set V' does not contain (δ, n)-heavy vertices as well as no edges of length at least δ. so it satisfies conditions (3), (4) in Lemma 4.8 . Finally condition (2) follows by the spreadness condition of feasible solutions. □

C.2. The oracle

We prove here Lemma 4.8. To simplify the description of the oracle, as in (Arora & Kale, 2007), we consider the following modification of Equation (3), which contains additional constraints. The two programs are equivalent as these constraints are *implied* by the ones in Equation (3).

$$\left\{\begin{array}{lll} \min\langle L, X\rangle & & \\ X_{ii} = 1 & \forall i \in [n] & \text{(unit norm)} \\ \langle T_p, X\rangle \geqslant 0 & \forall \text{paths } p & \text{(triangle inequality)} \\ \langle K, X\rangle \geqslant 4an^2 & & \text{(balance)} \end{array}\right\} \tag{8}$$

Remark C.6. We remark that, as we need not to explicitly write down the program, but only efficiently find separating hyperplanes, the use of Equation (8) does not imply an increase in the running time of the overall algorithm.

We consider the dual program of Equation (8), which has variables $x_1, \ldots, x_n$ for each vertex, f_p for every path p and an additional variable z for the set $[n]$ considered in the primal.

$$\left\{\begin{array}{ll} \max \sum_{i\in[n]} x_i + an^2 z & \\ \operatorname{diag}(x) + \sum_p f_p T_p + zK \preceq L & \\ f_p, z \geqslant 0 & \forall \text{ paths } p \end{array}\right\} \tag{9}$$

Now, given a candidate solution X, our starting point is the procedure of Lemma C.1, which we use to remove vertices that are heavy in the current embedding.

Proof of Lemma 4.8. Throughout the proof, whenever we write a feedback matrix, all the variables that are not specified are set to 0. Let X be the matrix denoting the current embedding. We may assume without loss of generality that both the oracles in Lemma 4.6 and Lemma 4.7 outputted **yes** on X. We claim there are at most $\bar{C} \cdot \alpha/\delta$ edges of length at least δ in the embedding for some large enough constant $\bar{C} > 0$. Suppose this is not the case, consider the following procedure:

- Pick uniformly at random an a-balanced bipartition A, B.
- Compute the max d-regular A-B flow with $d = O(\alpha)/n$.

In expectation the flow is larger than $\bar{C}\alpha$ (if it is smaller we have found a cut). If the flow is larger than $\bar{C}\alpha$ then let F be the Laplacian of the flow graph and let D be the Laplacian of the complete weighted graph where only edges ij with $i \in A\,, j \in B$ have weight f_{ij}, and the rest have 0 weight . Then by definition $\sum_p f_p T_p = F - D$. Thus we may set $x_i = \alpha/\left|V(G)\right|$ for all $i \in V(G)$, f_p as in the computed flow for all p and all other variables to 0. The feedback matrix Y becomes $\frac{\alpha}{|V(G)|}\mathrm{Id} + F - D - F = \frac{\alpha}{|V(G)|}\mathrm{Id} - D$ and we have

$$\langle \frac{\alpha}{|V(G)|}\mathrm{Id} - D, X\rangle \leqslant \alpha - (\bar{C} - 1)\cdot\alpha < -\alpha < 0\,.$$

Notice also that $\left\|\frac{\alpha}{|V(G)|}\mathrm{Id} - D\right\| \leqslant O(\alpha/\left|V(G)\right|)$. We repeat this procedure $O(\log n)^{100}$ times, by Markov's inequality the claim follows with probability at least $1 - O(\log n)^{-99}$. Let $E^* \subseteq E(G)$ be the set of edges of length at least δ in the embedding. Notice that we must have $|E^*| \leqslant O(\alpha/\delta)$.

Now we run the heavy vertex removal procedure Algorithm 4. Let (P_1, P_2, V') be the resulting partition. If the partition satisfies $(1), (2), (3), (4)$ in Lemma 4.8 the result follows. Else, since the oracles in Lemma 4.6 and Lemma 4.7 outputted **yes** on X, since there are no edges longer than δ in $E(G) \setminus E^*$ and since by construction V' does not contain (δ, n) heavy vertices, it must be that

$$\left|E(P_1, P_2, V') \cap \left\{ij \in E(G) \;\middle|\; \|v_i - v_j\|^2 \leqslant \delta\right\}\right| > C^* \cdot \frac{\alpha}{\delta} \cdot \left(1 + \frac{\ell}{\delta}\right)\,, \tag{10}$$

for some large enough constant $C^* > 10^{10}C$, where $C > 0$ is the universal constant of Lemma C.1.

Let $d = 100 \cdot C^* \cdot \left(\frac{\alpha}{|V(G)|\cdot\delta}\left(1+\frac{\ell}{\delta}\right)\right)$. We compute the maximum d-regular flows for each of the partitions $(P_1 \cup P_2\,, V')$, $(P_1\,, P_2 \cup V')$, $(P_2\,, P_1 \cup V')$ as described in Appendix A using the algorithm in Theorem A.1. By Equation (10) at least one of these cuts has flow $\frac{C^*}{3}\cdot\frac{\alpha}{\delta}\cdot\left(1+\frac{\ell}{\delta}\right)$ as otherwise by duality we have found a $(a/2)$-balanced cut of value at most $C^*\cdot\frac{\alpha}{\delta}\cdot\left(1+\frac{\ell}{\delta}\right)+|E^*| \leqslant O\left(\frac{\alpha}{\delta}\left(1+\frac{\ell}{\delta}\right)\right)$ as desired. Without loss of generality we may always assume this is the partition $(P_1 \cup P_2\,, V')$. We distinguish two cases:

1. $\sum_{ij\in E(P_1\cup P_2\,,V')\setminus E^*} f_{ij}\left\|v_i - v_j\right\|^2 \geqslant \frac{C^*}{10^9}\left(\alpha\left(1+\frac{\ell}{\delta}\right)\right)$,
2. $\sum_{ij\in E(P_1\cup P_2\,,V')\setminus E^*} f_{ij}\left\|v_i - v_j\right\|^2 < \frac{C^*}{10^9}\left(\alpha\left(1+\frac{\ell}{\delta}\right)\right)$.

Suppose we are in case 1. Let F be the Laplacian of the weighted graph corresponding to the flow and let D be the Laplacian of the complete weighted graph where only edges ij with $i \in P_1 \cup P_2$ and $j \in V'$ have weight f_{ij}, and the rest have 0 weight. Since we are in case 1 we have

$$\langle D, X\rangle \geqslant \frac{C}{10^9}\left(\alpha\left(1+\frac{\ell}{\delta}\right)\right).$$

Moreover, by definition $\sum_p f_p T_p = F - D$. Thus we set $x_i = \alpha/\left|V(G)\right|$ for all $i \in V(G)$, f_p as in the computed flow for all p and all other variables to 0. The feedback matrix Y becomes $\frac{\alpha}{|V(G)|}\mathrm{Id} + F - D - F = \frac{\alpha}{|V(G)|}\mathrm{Id} - D$ and we have

$$\langle \frac{\alpha}{|V(G)|}\mathrm{Id} - D, X\rangle \leqslant \alpha - \frac{C}{10^9}\cdot\alpha\left(1+\frac{\ell}{\delta}\right) < -\alpha < 0\,.$$

Moreover notice that $\left\|\frac{\alpha}{|V(G)|}\mathrm{Id} - D\right\| \leqslant O\left(\frac{\alpha}{|V(G)|}\right) + d \leqslant \tilde{O}\left(\frac{\alpha}{|V(G)|}\right)$ where in the last step we used the inequality $\delta \geqslant \Omega(1/\log n)$. In conclusion, in this case the ORACLE finds a separating hyperplane and outputs **no**. Notice also that by construction D has at most $O(m+n)$ non zero entries so the feedback matrix can be computed in time $O(m+n)$.

It remains to consider case 2. By Lemma C.1 and Markov's inequality, we know that for any feasible solution X^* to Equation (2) with objective value at most α, it holds with probability at least $1/2$:

$$\mathbb{E}\left[\left|E(P_1, P_2, V') \cap \left\{ij \in E(G) \;\middle|\; \left\|v_i - v_j\right\|^2 \leqslant \delta\right\}\right|\right] \leqslant C\cdot\frac{\alpha}{\delta}\cdot\left(1+\frac{\ell}{\delta}\right),$$

where $C < C^*/10^{10}$. Thus repeating the procedure $(\log n)^{100}$ times, we get that, for any feasible solution X^* with objective value α, with probability at least $1 - O(\log n)^{-100}$, for at least one of the resulting partitions $(P_1, P_2\,, V')$

$$\left|E(P_1, P_2\,, V') \cap \left\{ij \in E(G) \;\middle|\; \left\|v_i - v_j\right\|^2 \leqslant \delta\right\}\right| \leqslant C\cdot\frac{\alpha}{\delta}\cdot\left(1+\frac{\ell}{\delta}\right).$$

Now consider again our candidate solution X satisfying Equation (10). If after $(\log n)^{100}$ trials we still satisfy Equation (10) and are always in case 2, then with probability $1 - O(\log n)^{-100}$ there exist at least $\frac{C^*}{10}\cdot\frac{\alpha}{\delta}\cdot\left(1+\frac{\alpha}{\delta}\right)$ edges of length at most $\delta/10^8$ crossing the cut $E(P_1 \cup P_2\,, V') \setminus E^*$.

By design of Algorithm 4, then it must be the case that there exists a set of size $\Omega(n)$ of triplets $\{i,j,k\} \subseteq C$ with $ij \in E(G)$ and $k \in V^*$ such that $\left\|v_i - v_j\right\|^2 \leqslant \delta/10^8$, $\left\|v_j - v_k\right\|^2 \leqslant \left\|v_i - v_k\right\|^2$ but

$$\mathbb{P}_{r^{u.a.r.}\sim[1,2]}\left(\left\|v_k - v_j\right\|^2 \leqslant \delta(1+r) \text{ and } \left\|v_k - v_i\right\|^2 > \delta(1+r)\right) \geqslant 10^4\cdot\frac{\left\|v_i - v_j\right\|^2}{\delta}\,.$$

Indeed if this scenario does not apply then we would have seen a partition violating Equation (10) with probability at least $1 - O(\log n)^{-100}$ by the argument used in the proof of Lemma C.5.

So suppose this scenario applies, and consider such a triplet $\{i,j,k\}$. Then we must have

$$\left\|v_i - v_k\right\|^2 \geqslant \left\|v_j - v_k\right\|^2 + 10^4\cdot\left\|v_i - v_j\right\|^2 \tag{11}$$

so we are violating the triangle inequality. Furthermore, we know that the sum over each such triplets must satisfy

$$\sum_{\{i,j,k\} \text{ satisfying Equation (11)}} \|v_i - v_j\|^2 \geqslant \Omega\left(\frac{\alpha}{\delta}\left(1+\frac{\ell}{\delta}\right)\right).$$

as otherwise with probability $1 - O(\log n)^{-100}$ we would have found a cut violating Equation (10). Notice now that we can find such triangle inequalities in linear time by looking at the edges being cut and the vertices being picked at each iteration of Algorithm 4.

Thus set $x_i = \alpha/|V(G)|$ for all $i \in V(G)$ and $f_p = \frac{C^*\alpha}{n}$ for $\Theta(n)$ such violated triangle inequalities and a large enough constant $C^* > 0$. We set $F = \mathbf{0}$ and

$$\langle \frac{\alpha}{|V(G)|}\mathrm{Id} + \sum f_p T_p, X\rangle \leqslant \alpha - O\left(\frac{\alpha}{\delta}\right) \leqslant -\alpha < 0\,,$$

where in the last step we used the assumption $\delta < 1$. The width of the feedback matrix is at most $O(\alpha/|V(G)|)$ and it has $O(m+n)$ entries, thus it can be computed in $O(m+n)$ time.

Finally we remark that choosing $d = O(\log n)^{100}$ the oracle is d-robust by Lemma A.3.

□

D. The semi-random hierarchical stochastic model

In this section we consider the semi-random hierarchical stochastic model (HSM) from (Cohen-Addad et al., 2019) and develop a nearly linear time algorithm that estimates the Dasgupta's cost of the underlying hierarchical clustering model upto constant factor. The main idea is to recursively compute an $O(1)$-approximation to Balanced Cut which produces a graph with $O(1)$-approximation to the Dasgupta's cost (Dasgupta, 2016). Essentially most of this section is directly cited from (Cohen-Addad et al., 2019) and we only provide it for the completeness. However, note that using Theorem 1.2 we can improve the running time of the algorithm to the nearly linear time. In the following subsection, we formally define the Dasgupta's cost of the graph and the hierarchical stochastic model.

D.1. Related notions

Let $G = (V, E, w)$ be an undirected weighted graph with weight function $w : E \to \mathbb{R}^+$, where $\mathbb{R}^+$ denotes non-negative real numbers. For simplicity we let $w(x, y) = w(y, x) = w(\{x, y\})$. For set $U \subseteq V$ we define $G[U]$ to be the subgraph induced by U. A hierarchical clustering T of graph G is a rooted binary tree with exactly $|V|$ leaves, such that each leaf is labeled by a unique vertex $x \in V$.

For $G = (V, E)$ and a hierarchical-clustering tree T we denote the lowest common ancestor of vertex x and y in T by LCAT(x, y). For any internal node N of T, we let T_N to be the subtree of T rooted at N and we define $V(N)$ to be the set of leaves of the subtree rooted at N. Finally, for a weighted graph $G = (V, E, w)$ and any subset of vertices $A \subseteq V$ we define $w(A) = \sum_{x,y\in A} w(x, y)$, and for any set of edges $E0$, we let $w(E0) = \sum_{e\in E0} w(e)$.For any sets of vertices $A, B \subseteq V$, we also define $w(A, B) = \sum_{x\in A,y\in B} w(x, y)$.

Equipped with these notation we define the Dasgupta's cost of a graph for a tree as follows:

Definition D.1 ((Dasgupta's cost(Dasgupta, 2016; Cohen-Addad et al., 2019))). Dasgupta's cost of the tree T for the graph $G = (V, E, w)$ is defined as

$$\text{cost}(T; G) = \sum_{(x,y)\in E} \text{leaves}(T[LCA(x; y)]) \cdot w(x, y).$$

Definition D.2 (Ultrametric (Cohen-Addad et al., 2019)). A metric space (X, d) is an ultrametric if for every $x, y, z \in X$, $d(x, y) \leqslant \max\{d(x, z), d(y, z)\}$.

We say that a weighted graph $G = (V, E, w)$ is generated from an ultrametric if there exists an ultrametric (X, d), such that $V \subseteq X$, and for every $x, y \in V$, $x \neq y$, $e = \{x, y\}$ exists, and $w(e) = f(d(x, y))$, where $f : \mathbb{R}^+ \to \mathbb{R}^+$ is a non-increasing function. For a weighted undirected graph $G = (V, E, w)$ generated from an ultrametric, in general there may be several ultrametrics and corresponding functions f mapping distances in the ultrametric to weights on the edges, that generate the

same graph. It is useful to introduce the notion of a minimal ultrametric that generates G. Let (X, d) be an ultrametric that generates $G = (V, E, w)$ and f the corresponding function mapping distances to similarities. Then we consider the ultrametric $(V, \tilde{d})$ as follows: (i) $\tilde{d}(u, u) = 0$ and (ii) for $u \neq v$

$$\tilde{d}(u, v) = \tilde{d}(v, u) = \max_{u', v'} d(u', v') | f(d(u', v')) = f(d(u, v))$$

Definition D.3 (Generating Tree (Cohen-Addad et al., 2019))**.** Let $G = (V, E, w)$ be a graph generated by a minimal ultrametric (V, d). Let T be a rooted binary tree with $|V|$ leaves and $|V| - 1$ internal nodes; let N denote the internal nodes and L the set of leaves of T and let $\sigma : L \to V$ denote a bijection between the leaves of T and nodes of V . We say that T is a generating tree for G, if there exists a weight function $W : \mathcal{N} \to R^+$, such that for $N_1, N_2 \in \mathcal{N}$, if N_1 appears on the path from N_2 to the root, $W(N1) \leqslant W(N2)$. Moreover for every $x, y \in V$, $w(x, y) = W(\text{LCA}T(\sigma^{-1}(x), \sigma^{-1}(y)))$.

We say that a graph G is a ground-truth input if it is a graph generated from an ultrametric. Equivalently, there exists a tree T that is generating for G.

Now we are ready to define Hierarchical Stochastic Model graphs as follows:

Definition D.4 (Hierarchical Stochastic Model (HSM) (Cohen-Addad et al., 2019))**.** . Let $\widetilde{T}$ be a generating tree for an n-vertex graph $\bar{G}$, called the expected graph, such that all weights are in $[0, 1]$. A hierarchical stochastic model is a random graph G such that for every two vertices u and v, the edge $\{u, v\}$ is present independently with probability $w(\{u, v\}) = W(\text{LCA}_T(\sigma^{-1}(u), \sigma^{-1}(v)))$, where w and W are the weights functions associated with $\widetilde{T}$ as per Definition D.3.

In other words, the probability of an edge being present is given by the weight of the lowest common ancestor of the corresponding vertices in $\widetilde{T}$.

D.2. The algorithm for the semi-random hierarchical stochastic model

We generate a random graph, $G = (V, E)$, according to HSM (Definition D.4). The semi-random model considers a random HSM graph generated as above where an adversary is allowed to only remove edges from G. Note that the comparison is to the cost of the generating tree on the graph $\bar{G}$ (Definition D.4). In this section we present the proof of Theorem 1.3.

Proof of Theorem 1.3 is a variant of Theorem 6.1 from (Cohen-Addad et al., 2019) with nearly-linear running time that uses our fast algorithm for finding Balanced-Cut.

Let $\bar{G}_n = (\bar{V}_n, \bar{E}_n, w)$ be a graph generated according to an ultrametric, where for each $e \in \bar{E}_n$, $w(e) \in (0, 1)$ (Definition D.3). Let $G = (V, E)$ be an unweighted random graph with $|V| = \left|\bar{V}_n\right| = n$ generated from $\bar{G}$ as follows. For every $u, v \in \bar{V}_n$ the edge (u, v) is added to G with probability $w((u, v))$ (Definition D.4).

We assume that $V = \bar{V}_n$ and let T be a generating tree for $\bar{G}$. Let $U \subseteq V$. Let $\widetilde{T}|_U$ denote the restriction of T to leaves in U Let $N(U)$ be the root of $\widetilde{T}|_U$. Consider the following procedure where the nodes appear as leaves in the left and right subtrees of the root of $\widetilde{T}|_U$. Suppose we follow the convention that the left subtree is never any smaller than the right subtree in $\widetilde{T}|_U$. We say that the canonical node of $\widetilde{T}|_U$ is the first left node N_L encountered in a top-down traversal starting from $N(U)$ such that $(1 - b) \cdot |U| \geqslant V(N_L) \geqslant b \cdot |U|$, where, $0 < b < 1/2$ is a constant. We define $U_L = V(N_L)$, and $U_R = U \setminus U_L$. We say that (U_L, U_R) is the *canonical* cut of U. It is easy to see that such a cut always exists since the tree is binary and left subtrees are never smaller than right subtrees. Let $E_{rnd} = \{(u, v) \in E | u \in U_L, v \in U_R\}$.

Lemma D.5 ((Cohen-Addad et al., 2019))**.** *For a random graph G generated as described in Theorem 1.3, with probability at least $1 - o(1)$, for every subset U of size at least $n^{2/3}\sqrt{\log n}$, the subgraph (U, E_{rnd}) is geometrically expanding up to scale $(1/\sqrt{D}, n, \alpha)$ where*

$$\alpha = C. \max\{w(L, R), |U| \cdot D \cdot \log^2 D, |U| \cdot D \cdot \log n\}, \tag{12}$$

Furthermore, the result also applies in the semi-random setting where an adversary may remove any subset of edges from the random graph G.

Theorem D.6. *For any graph $G = (V, E)$, and weight function $w : E \to \mathbb{R}^+$, the ϕ-sparsest-cut algorithm from (Cohen-Addad et al., 2019) outputs a solution of cost at most $O(\phi \cdot OPT)$.*

Now we show the proof of Theorem 1.3 which is a variant of Theorem 6.1 from (Cohen-Addad et al., 2019) using our nearly-linear time agorithm for Balanced-Cut.

Proof of Theorem 1.3. Let $b = 1/3$. By Theorem D.6, the recursive sparsest cut algorithm approximates Dasgupta's cost up to factor $O(\phi)$ assuming that at every recursion step, the algorithm is provided with a ϕ-approximation to the b-Balanced Cut problem (i.e., minimize cut subject to the constraint that both sides have at least b fraction of vertices being cut)

Note that $cost(\widetilde{T}; \bar{G}) = \Omega(n^3 \cdot p_{min}) = \Omega\left(n^{7/3} \cdot \log n\right)$. Therfore, once we obtain sets U of size $(n_0 = n^{2/3} \cdot \log n)$, since there are at most n/n_0 of them, even if we use an arbitrary tree on any such U, together this can only add $O(\frac{n}{n_0} \cdot n_0^3) = O(n^{13/9} \cdot (\log n)^2) = O\left(n^{7/3} \cdot \log n\right)$ to the cost. Thus, we only need to obtain suitable approximations during the recursive procedure as long as $|U| \geqslant n^{2/3} \cdot \log n$. This is precisely given by using Lemma D.5. Let $D = O(\log n)$, $\delta = \frac{1}{\sqrt{D}} = O\left(\frac{1}{\sqrt{\log n}}\right)$, let $\kappa \geqslant \Omega(\sqrt{\log n})$. Observe that in Equation 12, $w(L, R) = \Omega(|U|^2 \cdot p_{min}) = \Omega\left(n^{2/3}(\log n)^3\right)$, $|U| D \log^2 D = o(|U| D \log n)$, and $D|U| \log n = O\left(n^{2/3}(\log n)^3\right)$. Let $\alpha = O(w(L, R))$. Thus, by Theorem 4.3 there exists an algorithm that runs in time $\tilde{O}\left(|V(G)| + |E(G)|\right)$ and returns a cut that is an approximation to the $\Omega(b)$-balanced partition (S, T) with cut of size $|E(S, T)| \leqslant O(\alpha \cdot (1 + \delta \cdot \kappa \cdot \sqrt{\log n})) = O(\alpha)$ on the induced subgraph of $\bar{G}$ on the vertex set U. This observation together with the case where subgraphs have size less than $n^{2/3} \log n$ finishes the proof. □