Non-Uniform Graph Partitioning

Robert Krauthgamer[*] Joseph (Seffi) Naor[†] Roy Schwartz[‡] Kunal Talwar[§]

Abstract

We consider the problem of NON-UNIFORM GRAPH PARTITIONING, where the input is an edge-weighted undirected graph $G = (V, E)$ and k capacities $n_1, \ldots, n_k$, and the goal is to find a partition $\{S_1, S_2, \ldots, S_k\}$ of V satisfying $|S_j| \le n_j$ for all $1 \le j \le k$, that minimizes the total weight of edges crossing between different parts. This natural graph partitioning problem arises in practical scenarios, and generalizes well-studied balanced partitioning problems such as MINIMUM BISECTION, MINIMUM BALANCED CUT, and MINIMUM k-PARTITIONING. Unlike these problems, NON-UNIFORM GRAPH PARTITIONING seems to be resistant to many of the known partitioning techniques, such as spreading metrics, recursive partitioning, and Räcke's tree decomposition, because k can be a function of n and the capacities could be of different magnitudes.

We present a bicriteria approximation algorithm for NON-UNIFORM GRAPH PARTITIONING that approximates the objective within $O(\log n)$ factor while deviating from the required capacities by at most a constant factor. Our approach is to apply stopping-time based concentration results to a simple randomized rounding of a configuration LP. These concentration bounds are needed as the commonly used techniques of bounded differences and bounded conditioned variances do not suffice.

1 Introduction

Graph partitioning problems have been studied extensively in the last few decades, exhibiting beautiful connections to metric geometry, functional analysis, and computational complexity, including e.g. [LR99, LLR95, AR98, ENRS99, FK06, RÖ8, ARV08, KNS09, CGR08, ALN08, LN06, CKN09] in the case of approximation algorithm for balanced partitioning. Typically, the goal is to partition an input graph into several parts, so as to minimize the cost of the cut, i.e., the total weight of edges connecting different parts. Such problems arise in many diverse settings, including parallel and cloud computing, data mining and clustering, pattern recognition, VLSI layout design, and sparse linear systems.

In balanced partitioning, one seeks to break the graph into two parts of equal size (as in MINIMUM BISECTION and its variant BALANCED b-CUT), or into k equal-size parts (as in MINIMUM k-PARTITIONING). These problems are *uniform* in the sense that the requirement on the parts' sizes is symmetric and does not depend on the identity of each part. But recently, there is a growing interest in non-uniform graph partitioning, including SMALL-SET EXPANSION [RS10, RST10, RST12, BFK+11], and SPARSEST k-PARTITIONING [LRTV11, LOT12, LRTV12, KLL+13, LM13].

We consider the NON-UNIFORM GRAPH PARTITIONING problem, whose input is an undirected graph $G = (V, E)$ equipped with edge weights $w : E \to \mathcal{R}_+$ and k capacities $n_1 \ge n_2 \ge \ldots \ge n_k \ge 1$, and the goal is to find a partition $\{S_1, S_2, \ldots, S_k\}$ of V satisfying the capacities, namely, $|S_j| \le n_j$ for all $1 \le j \le k$, that minimizes the cost $\sum_{i<j} \delta(S_i, S_j)$ (the total weight of edges crossing between different parts in the partition). To ensure feasibility, the sum of the capacities is $\sum_{j=1}^{k} n_j \ge n$, where $n = |V|$ is the number of vertices in the graph G. We stress that the number of capacities k can be a function of n and that the capacities could possibly be of significantly different magnitudes.

The NON-UNIFORM GRAPH PARTITIONING problem is a natural generalization of well-known cut problems like MINIMUM BISECTION [FK06, RÖ8], BALANCED b-CUT [LR99, ARV08] and MINIMUM k-PARTITIONING [ENRS99, KNS09]. This problem not only arises in many of the above mentioned settings (e.g. parallel computing, clustering, and VLSI layout design), but it has additional applications. For example, the design of bandwidth-constrained datacenters requires one to solve NON-UNIFORM GRAPH PARTITIONING [BMC+12] (consult the references therein for more details). Due to the wide range of applications, a sequence of works, which studies heuristics for solving NON-UNIFORM GRAPH PARTITIONING, exists [Bar82, BVW88, San89, HMV92, RW95]. These heuristics utilize different tools such as spectral theory and quadratic programming. Unfortunately, to the best of our knowledge, no provable guarantee is known for the NON-UNIFORM GRAPH PARTITIONING problem.

[*]Weizmann Institute of Science, Rehovot, Israel. Work supported in part by the Israel Science Foundation (grant #897/13), the US-Israel BSF (grant #2010418), and by the Citi Foundation. Email: robert.krauthgamer@weizmann.ac.il

[†]Technion - Israel Institute of Technology, Haifa, Israel. Work supported by ISF grant 954/11 and BSF grant 2010426. Email: naor@cs.technion.ac.il

[‡]Microsoft Research. Email: roysch@microsoft.com

[§]Microsoft Research. Email: kunal@microsoft.com

1.1 Our Results.

We present an efficient algorithm that achieves an approximation factor of $O(\log n)$ while violating the capacity of each part by at most a constant factor.

THEOREM 1.1. *There are absolute constants $c, C \geq 1$ and a polynomial-time algorithm, which given as input an edge weighted graph G and k capacities $n_1, \ldots, n_k$, outputs a partition $\{S_1, \ldots, S_k\}$ of V such that $|S_i| \leq cn_j$ for all $1 \leq j \leq k$ and*

$$\sum_{i<j} \delta(S_i, S_j) \leq C \log n \sum_{i<j} \delta(S_i^*, S_j^*)$$

where $\{S_1^*, \ldots, S_k^*\}$ is an optimal partition for NON-UNIFORM GRAPH PARTITIONING.

We defer the analysis that optimizes the constant c (the multiplicative capacity violation) to the full version of this paper.

1.2 Techniques.

Even though NON-UNIFORM GRAPH PARTITIONING is closely related to many balanced graph partitioning problems, known approaches for the latter fail when applied to NON-UNIFORM GRAPH PARTITIONING. First, the natural formulation of a spreading metric [ENRS99, ENRS00] for the problem is not strong enough, since (by convexity) it admits a fractional solution in which all vertices are spread equally with respect to the *average* capacity. When the capacities $n_1, \ldots, n_k$ vary in magnitude significantly, such an average spreading guarantee is too weak to represent parts whose sizes are close to the required ones. Second, the application of a recursive partitioning approach, which gradually decomposes the graph into parts of the desired sizes, fails because we have no method to correct mistakes at intermediate steps, and reverting them leads to inefficient backtracking. Third, reducing the problem to a tree via Räcke [RÖ8] followed by dynamic programming is not effective as the number of capacities k might depend on n, and the capacities $n_1, \ldots, n_k$ can have vastly different magnitudes. Hence, known methods like [AR06, FF12] do not yield polynomial runtime. Even if one approximates the capacities, say by rounding them to powers of 2, there could easily be $\log k$ distinct capacity values, requiring $n^{O(\log k)}$ runtime.

We overcome these obstacles by employing the following three techniques.

Instance Simplification. We simplify the capacities sequence, which is crucial in order to bound the capacity violation of our algorithm. First, we round the capacities $n_1, \ldots, n_k$ into powers of 2. Second, the rounded capacities are increased so that the "aggregate" capacity is geometrically increasing; more precisely, a strict decrease in capacity, i.e., $n_i > n_{i+1}$, means that the aggregate of all capacities bigger or equal to n_i is significantly smaller than the aggregate of capacities bigger or equal to n_{i+1}. Of course, we must be able to revert the process in the sense of mapping a solution to the simplified instance back to a solution to the original instance, while violating the capacity constraints by at most another constant factor.

Configuration LP. We take a combinatorial approach and write a configuration LP for choosing (fractionally) a collection of vertex subsets (interpreted as cuts) whose sizes meet the capacity constraints, and altogether they cover all vertices of the graph. As this LP contains an exponential number of variables, an efficient method for solving it is needed.

The standard argument used in such cases, solving the dual of the configuration LP, does not seem to work as the LP contains *both* packing and covering constraints and we have only an *approximate* dual separation oracle. To overcome this, we reduce the configuration LP to a decision problem of whether a given polytope $\mathcal{Q}$ is empty. Techniques designed for packing-covering polytopes, such as $\mathcal{Q}$, when applied as a black box, ultimately result in a loss of $O(\log n)$ in both the objective's value and in the capacity violation factor ($O(\log n)$ is the approximation factor of our dual separation oracle). A careful application of these techniques that uses a different scaling factor for different constraints, results in a loss of $O(\log n)$ only in the objective value of the configuration LP, whereas there is no such loss in any of the LP constraints.

Measure Concentration via Stopping Times. Given a solution to the configuration LP, we employ a simple probabilistic rounding that repeatedly chooses a vertex subset (cut) with probability proportional to its (fractional) value in the LP, until all vertices are assigned. Bounding the cost of such a randomized partitioning is straightforward – this rounding increases the cost only by a factor of 2 compared to the LP. However, bounding the total number of vertices assigned to a part in the partition is not immediate.

The obvious approach for bounding the probability of having too many vertices in some part is to use Markov's inequality and apply a union bound over the different capacities. This results in a capacity violation of $O(k)$, a completely useless guarantee as the capacity of the largest piece is at least n/k. To overcome this, we observe that parts of approximately equal capacities are interchangeable. One could therefore clump the capacities (which we call "buckets") into "mega-buckets" and consider the total number of vertices assigned to this mega-bucket. The above argument, applied now for the mega-buckets, gives an improved $O(\log k)$ deviation form the capacity, as one can argue using pruning arguments that there are at most $O(\log k)$ mega-buckets.

Another natural approach is to bound the capacity deviation by relying on measure concentration results, and therefore the next thing to consider is the Doob martingale that corresponds to the total number of vertices assigned to a mega-bucket. When usually applying martingale concentration results one of the following two is used, an absolute upper bound on the martingale's difference in two consecutive steps (also known as the Lipschitz condition) or an upper bound on the variance of this difference conditioned on the *worst* possible history (also known as the variance condition). Unfortunately, these two upper bounds could equal the expected total number of vertices assigned to the mega-bucket, and thus accumulate rapidly in each step of the rounding. Since the rounding process needs to make $\Omega(k \log n)$ steps until all vertices are chosen, the above two concentration techniques provide an overall $O(\sqrt{k \log n \log\log k})$ multiplicative violation of the capacities, an even worse violation than the previous $O(\log k)$[1].

To bypass this issue, we take a different approach and prove that with high probability the sum of the conditioned variances is sufficiently small. We stress that in contrast to the concentration result based on the variance condition, where in each step a possibly different worst history is condition on, we prove that the random variable which equals the sum of the conditioned variances is sufficiently small with good probability. Stopping-time based concentration results are now used along with this fact, to obtain on overall capacity deviation of $O(\log\log k)$. So how then do we get only a constant violation? This is where the instance simplification comes in. We show that if the buckets satisfy a certain volume condition then the probabilities of constant deviation form a geometric sequence, hence a union bound over the mega-buckets suffices to achieve a bounded failure probability. The instance simplification done initially ensures that the mega-buckets' aggregates increase geometrically, and therefore the required volume condition is satisfied.

1.3 Related Work. In [LR99] Leighton and Rao gave an $O(\log n)$ approximation for SPARSEST CUT which is based on a linear spreading metric, resulting in a bi-criteria $O(\log n)$ approximation for BALANCED b-CUT. This was subsequently improved to an $O(\sqrt{\log n})$ guarantee via the use of negative type spreading metrics for both problems, by Arora, Rao and Vazirani [ARV08]. Using the negative type spreading metric machinery of the latter, Chawla et. al [CGR08] presented an approximation of $O(\log^{3/4} n)$ for the general demands version of SPARSEST CUT. This was improved by Arora et. al. [ALN08] to $O(\sqrt{\log n}\log\log n)$ who constructed a better embedding of negative type metrics into Euclidean space. For the MINIMUM BISECTION problem, Feige and Krauthgamer [FK06] provided a true approximation with an approximation guarantee of $O(\log^{3/2} n)$, which was improved by Räcke [RÖ8] to $O(\log n)$ by constructing a distribution over trees that approximates all cuts in the graph. Makarychev et. al [MMV12] consider the SPARSEST CUT problem in the semi-random model and provide improved guarantees for it. If one wishes to partition a graph into k pieces of equal size, Even et. al. [ENRS99] provided a bi-criteria $O(\log n)$ algorithm by using the region growing technique of Garg et. al. [GVY93]. Building upon the negative-type spreading metric machinery of [ARV08], Krauthgamer et. al. [KNS09] improved the latter bound to $O(\sqrt{\log k \log n})$.

A different related work is that of Andrews et. al. [AHKM11] for the CAPACITATED METRIC LABELING problem. This problem, when the metric is uniform, is identical to NON-UNIFORM GRAPH PARTITIONING with additional assignment costs specifying how much it costs assigning a vertex to a piece in the partition. [AHKM11] proved, assuming $NP \not\subseteq ZPTIME(n^{polylog(n)})$, that there is no algorithm with a finite approximation guarantee for CAPACITATED METRIC LABELING that violates the capacities by a factor of $o(\sqrt{\log k})$. This should be in contrast to the constant capacity violation of our algorithm, which heavily uses the fact that any vertex can be assigned to any piece, i.e., all assignment costs are 0.

Paper Organization. The instance simplification reduction appears in Section 2, whereas the configuration LP is in Section 3. Our randomized rounding algorithm and our main technical argument of concentration via stopping times appear in Section 4. Section 5 contains the algorithm for approximately solving the configuration LP.

2 Instance Simplification

We first show that an instance of NON-UNIFORM GRAPH PARTITIONING can be reduced to a different one in which the capacities, also called henceforth *buckets*, satisfy some special properties. Given an instance $(G = (V, E), C = \{n_1, \ldots, n_k\})$ of NON-UNIFORM GRAPH PARTITIONING, we will simplify C to have a special structure. We introduce some definitions first.

DEFINITION 2.1. *Let $C = \{n_1, \ldots, n_k\}$ and let $C' = \{m_1, \ldots, m_{k'}\}$. We say that $C \preceq C'$ if there exists a mapping $f : [k] \to [k']$ such that*

$$\forall j \in [k'] \sum_{i \in [k]: f(i) = j} n_i \le m_j \ .$$

PROPOSITION 2.1. *Given a solution $\mathcal{S}$ to (G, C) and a mapping f that corresponds to $C \preceq C'$, one can construct a solution $\mathcal{S}'$ to (G, C') such that $cost(\mathcal{S}') \le cost(\mathcal{S})$.*

Proof. Let f be the mapping certifying $C \preceq C'$ and let vertex v be assigned to bucket i in $\mathcal{S}$. Then in $\mathcal{S}'$, we assign v to bucket $f(i)$. It is easy to see that the cost of $\mathcal{S}'$ is at most that of $\mathcal{S}$. Moreover, the definition of the $\preceq$ relation implies that if $\mathcal{S}$ was feasible, then so is $\mathcal{S}'$. □

[1]One can make only $O(k \log k)$ steps to cover all vertices except a fraction of $1/poly(k)$ which can be assigned to the largest mega-bucket. This results in a slightly better overall deviation of $O(\sqrt{k \log k \log\log k})$.

The following lemma argues that rounding bucket sizes to the next power of two gives us an instance which is essentially equivalent up to a factor of two in the capacities. We use αC to denote the scaling of each capacity in a multiset C by a factor $\alpha > 0$.

LEMMA 2.1. *Let $C = \{n_1, \ldots, n_k\}$ and let $C' = \{m_1, \ldots, m_k\}$ where each $m_i = 2^{\lceil \log_2 n_i \rceil}$ is derived by rounding n_i up to the next power of two. Then $C \preceq C' \preceq 2C$.*

Proof. The identity map certifies the domination in both directions. □

Let $C = \{n_1, n_2, \ldots, n_k\}$ be an instance in which all buckets sizes are powers of two and let $n_1 \geq n_2 \geq \ldots \geq n_k$. Let k_j denote the number of occurrences of $2^{-j} n_1$ in C. We call this set of k_j buckets a *megabucket*, and let its *volume*, denoted by $\mathrm{vol}(j)$ be defined as the total capacity of buckets in it; thus $\mathrm{vol}(j) = k_j 2^{-j} n_1$.

The following sufficient condition for dominance will be useful.

LEMMA 2.2. *Let $C = \{n_1, \ldots, n_k\}$ and $C' = \{n'_1, \ldots, n'_{k'}\}$ be such that all bucket sizes are powers of two, $n_1 \geq n_2 \geq \ldots \geq n_k$, $n'_1 \geq n'_2 \geq \ldots \geq n'_{k'}$, and let $\mathrm{vol}(j)$ and $\mathrm{vol}'(j)$ denote the respective megabucket volumes. Suppose that $n_1 = n'_1$ and for all $p \geq 0$, $\sum_{j \leq p} \mathrm{vol}(j) \leq \sum_{j \leq p} \mathrm{vol}'(j)$. Then $C \preceq C'$.*

Proof. We define the mapping f greedily: with n_i's and n'_i's sorted in decreasing order, we match each $i \in [k]$ to the first bucket whose capacity has not been filled already; i.e. having defined f on $[i-1]$, we define $f(i)$ to be the smallest i' such that $\sum_{j<i: f(j)=i'} n_j < n'_{i'}$. The volume condition ensures that $n'_{i'} \geq n_i$, and moreover, since all n_i's are powers of two, the remaining capacity in $n'_{i'}$ must have been at least n_i, and hence the assignment of $f(i)$ does not violate the capacity of i'. Continuing in this manner, we can define an f certifying the dominance. The claim follows. □

We next introduce the notion of a *nice instance*, which informally means that all bucket sizes are powers of two, and each non-empty megabucket (i.e. the set of buckets of size 2^j for some j) has volume at least a factor of $b > 1$ larger than the volume of every megabucket for size larger than 2^j.

DEFINITION 2.2. (NICE INSTANCE) *A set of capacities $C = \{n_1, n_2 \ldots, n_k\}$ with $n_1 \geq n_2 \geq \ldots \geq n_k$ is called b-nice if there is a sequence of integers $0 = p_1, \ldots, p_l$ such that:*

- *n_1 is a power of two;*
- *all bucket sizes n_i belong to the set $\{2^{-p_j} n_1 : j \in [l]\}$;*
- *let k_j be the number of occurrences of $2^{-p_j} n_1$ in C. Then the k_j's satisfy*

$$k_j 2^{-p_j} n_1 \geq b \cdot k_{j-1} 2^{-p_{j-1}} n_1 \quad \forall j > 1$$

The following is the central claim of this section, establishing that any set of capacities C is well-approximated by one that is nice.

THEOREM 2.1. *For any set of capacities $C = \{n_1, \ldots, n_k\}$, and for any constant $b \geq 1$, one can construct a set of b-nice capacities C' such that*

$$C \preceq C' \preceq c \cdot C,$$

where $c \leq O(b)$.

Proof. By lemma 2.1, we can assume that all n_i's are powers of two; this will affect the constant c in the theorem by at most a factor of two.

Informally, we will go down the list of megabuckets in decreasing order of bucket sizes, and delete megabuckets until we have deleted much more volume than the last non-empty megabucket, at which point we add the deleted volume to the megabucket at which we stop. We repeat this until there are no more megabuckets left, and increase the capacity of all megabuckets to restore dominance. We next formalize this intuition.

We start with $p_1 = 0$ and let $\mathrm{nvol}(1)$ denote the volume of the first megabucket. We add to C' $(b+1) \cdot k_1$ buckets of size n_1. Given p_j and $\mathrm{nvol}(j)$ we set p_{j+1} to be the smallest value (if any) such that $\sum_{i=p_j+1}^{p_{j+1}} \mathrm{vol}(i)$ is at least $b \cdot \mathrm{nvol}(j)$. We then set $\mathrm{nvol}(j+1) \triangleq \sum_{i=p_j+1}^{p_{j+1}} \mathrm{vol}(i)$, and add $(b+1) \cdot \mathrm{nvol}(j+1)/(2^{-p_{j+1}} n_1)$ buckets of size $2^{-p_{j+1}} n_1$ to C'. We repeat this until we run out of megabuckets.

The definition of C' ensures that it is b-nice. We next argue that $C \preceq C' \preceq c \cdot C$. To certify these relations, we use Lemma 2.2.

First, we claim inductively that for all $p \geq 1$, $\sum_{i \leq p} \mathrm{vol}(i) \leq \sum_{i \leq p} \mathrm{vol}'(i)$. Since $\mathrm{vol}'(i)$ is non-zero only when $i = p_j$ for some j, it suffices to prove the claim for all $p = p_j - 1$. For $p = p_2 - 1$, we have

$$\begin{aligned}\sum_{i \leq p} \mathrm{vol}(i) &= \mathrm{vol}(0) + \sum_{i=1}^{p_2 - 1} \mathrm{vol}(i) \\ &\leq \mathrm{vol}(0) + b \cdot \mathrm{vol}(0) \\ &= (b+1)\,\mathrm{vol}(0) = \mathrm{vol}'(0)\end{aligned}$$

where the inequality follows from the definition of p_2. Assuming the claim is true for $p = p_j$, we write

$$\begin{aligned}\sum_{i \le p} \mathrm{vol}(i) &= \sum_{i \le p_j - 1} \mathrm{vol}(i) + \mathrm{vol}(p_j) + \sum_{i=p_j+1}^{p} \mathrm{vol}(i) \\ &\le \sum_{i \le p_j - 1} \mathrm{vol}'(i) + (b+1)\,\mathrm{vol}(p_j) \\ &= \sum_{i \le p_{j+1}-1} \mathrm{vol}'(i)\end{aligned}$$

This proves that $C \preceq C'$.

We next prove the other direction. Since $\mathrm{vol}'(i)$ is nonzero only when $i = p_j$ for some j, it suffices to prove the claim for all $p = p_j$. For $p = p_0$, the definition of C' implies the claim for $c = (b+1)$. For $j \ge 0$, the added vol' at $p = p_{j+1}$ is by definition equal to $(b+1)\sum_{i=p_j+1}^{p_{j+1}} \mathrm{vol}(i)$, which is exactly $(b+1)$ times the added vol from $p_j + 1$ to p_{j+1}. Thus the induction holds and we conclude that $C' \preceq (b+1)C$.

The assumption that all bucket sizes in C were powers of two costs us another factor of two, so overall the theorem holds with $c = 2(b+1)$. □

3 Configuration LP

Denote the collection of all possible cuts (vertex subsets) of size n_j by $\mathcal{F}_j \triangleq \{S : S \subseteq V, |S| \le n_j\}$. Consider the following configuration LP:

$$\begin{array}{llll} (\mathcal{P}) & \min & \frac{1}{2}\sum_{j=1}^{k}\sum_{S\in\mathcal{F}_j} \delta(S)\cdot x_{S,j} & \\ (3.1) & \text{s.t.} & \sum_{j=1}^{k}\sum_{S\in\mathcal{F}_j : u\in S} x_{S,j} \ge 1 & \forall u \in V \\ (3.2) & & \sum_{S\in\mathcal{F}_j} x_{S,j} \le 1 & \forall j = 1,\dots,k \\ & & x_{S,j} \ge 0 & \forall j = 1,\dots,k, \forall S \in \mathcal{F}_j \end{array}$$

It is easy to check that the above linear program is a relaxation for NON-UNIFORM GRAPH PARTITIONING. In Section 5 we show that this linear program can be approximately solved in polynomial time. Specifically, we prove in Section 5 the following theorem.

THEOREM 3.1. *For every constant $0 < \varepsilon < 1$ there is a polynomial-time algorithm that, given as input an instance of $(\mathcal{P})$ represented succinctly using the graph G and the sizes $n_1, \dots, n_k$, outputs $\vec{x}$ such that:*

$$\begin{array}{ll} \frac{1}{2}\sum_{j=1}^{k}\sum_{S\in\mathcal{F}_j} \delta(S)\cdot x_{S,j} \le O(\log n)\cdot \mathrm{OPT}(\mathcal{P}) & \\ \sum_{j=1}^{k}\sum_{S\in\mathcal{F}_j : u\in S} x_{S,j} \ge 1, & \forall u \in V \\ \sum_{S\in\mathcal{F}_j} x_{S,j} \le 1 + \varepsilon, & \forall j = 1,\dots,k \\ x_{S,j} \ge 0. & \forall j = 1,\dots,k, \forall S \in \mathcal{F}_j \end{array}$$

Here $\mathrm{OPT}(\mathcal{P})$ is the optimum (objective value) of $(\mathcal{P})$.

Note that a solution $\vec{x}$ obtained from the above theorem is feasible except for constraint (3.2), which might be violated by a multiplicative constant of $1 + \varepsilon$. In order to make x feasible, we change the instance to a new one in which all capacities are doubled. This can be done as follows. Choose ε such that $1 + \varepsilon \le 3/2$ and for every bucket j partition $X = \{S : x_{S,j} > 0\}$ into two sets A and B such that $\sum_{S\in A} x_{S,j} \le 1$ and $\sum_{S\in B} x_{S,j} \le 1$ (start with $A = B = \emptyset$ and go over the elements of X in a non-increasing order of $x_{S,j}$ and assign each cut S to A if $\sum_{S\in A} x_{S,j} \le \sum_{S\in B} x_{S,j}$ or to B otherwise). Therefore, up to an additional loss of two in the capacity violation, from this point on we assume the we are given a feasible fractional solution for $(\mathcal{P})$ whose cost is at most $O(\log n)\cdot \mathrm{OPT}(\mathcal{P})$.

4 Randomized Rounding

In order to simplify the presentation of the rounding algorithm and its analysis, as mentioned before, we refer to each capacity n_j as a *bucket* that contains cuts. We assume that the instance is b-nice for a suitable constant b to be determined later. As before, we aggregate the buckets into *mega-buckets* according to powers of 2 as follows:

$$W_i \triangleq \left\{ j : n_j = 2^{-(i-1)} n_1 \right\} \qquad \forall i = 1,\dots,\ell\,.$$

Denote by k_i the number of buckets in the ith mega-bucket, namely $k_i \triangleq |W_i|$, $\forall i = 1,\dots,\ell$. Given the jth bucket, $1 \le j \le k$, let $m(j)$ be the mega-bucket to which the jth bucket belongs, namely $j \in W_{m(j)}$.

4.1 Algorithm. Informally, the algorithm repeatedly picks a bucket j uniformly at random, picks an S according to the distribution given by $x_{S,j}$, and assigns the yet-unassigned portion of S to megabucket $m(j)$. Once all vertices have been assigned, we assemble the pieces assigned to each megabucket in the appropriate number of pieces. We will later argue that if the instance is nice, this can be done while violating the capacity of each bucket by at most a constant factor. We next describe the algorithm in more detail.

The algorithm's description uses the following notations: t is the iteration index, $\mathcal{V}$ is the set of vertices covered by the algorithm so far, $\mathcal{W}_i$ is the collection of cuts (vertex subsets) assigned by the algorithm to the i-th mega-bucket W_i, J_t is the random bucket chosen by the algorithm in iteration t, Y_t is the random cut chosen by the algorithm in iteration t from bucket J_t, $Z_t \subseteq Y_t$ is the set of uncovered vertices in Y_t, and $m(u)$ is the mega-bucket which vertex u was assigned to by the algorithm.

```
1: $t \leftarrow 1$ and $\mathcal{V} \leftarrow \emptyset$.
2: $\mathcal{W}_i \leftarrow \emptyset$ for all $i = 1, \dots, \ell$.
3: while $\mathcal{V} \neq V$ do
4:     pick $J_t \in \{1, \dots, k\}$ uniformly at random.
5:     pick a cut $Y_t$ at random
         (where $\Pr[Y_t = S] = x_{S,J_t}\ \forall S \in \mathcal{F}_{J_t}$).
6:     $Z_t \leftarrow Y_t \setminus \mathcal{V}$.
7:     $m(u) \leftarrow m(J_t)$ for all $u \in Z_t$.
8:     $\mathcal{W}_{m(J_t)} \leftarrow \mathcal{W}_{m(J_t)} \cup \{Z_t\}$ and $\mathcal{V} \leftarrow \mathcal{V} \cup Z_t$.
9:     $t \leftarrow t + 1$.
10: for $i = 1, \dots, \ell$ do
11:     while $\mathcal{W}_i$ contains more than $k_i$ cuts do
12:         merge the smallest two cuts in $\mathcal{W}_i$.
13: return the partition $\cup_{i=1}^{\ell} \mathcal{W}_i$.
```

One should note that there might be an iteration t in which $Y_t = \emptyset$. This can happen only when $\sum_{S \in \mathcal{F}_{J_t}} x_{S,J_t} < 1$, and in this case with a probability of $1 - \sum_{S \in \mathcal{F}_{J_t}} x_{S,J_t}$, the algorithm chooses $Y_t = \emptyset$ in step (5).

4.2 Cost Analysis. Let $p_{u,v}$ denote the probability that u and v are assigned (to cuts) in different iterations of the algorithm, formally

$$p_{u,v} \triangleq \Pr\left[\exists t \neq t' : u \in Z_t, v \in Z_{t'}\right].$$

Obviously, $p_{u,v}$ is an upper bound on the probability that u and v are separated in the algorithm's output.

LEMMA 4.1. *For every $u, v \in V$ we have*

$$p_{u,v} \leq \sum_{j=1}^{k} \sum_{S \in \mathcal{F}_j : (u,v) \in \delta(S)} x_{S,j}.$$

Proof. Let us say that a pair (u, v) gets *settled* in iteration t if neither of u and v has been assigned by iteration $t-1$, but at least one of them gets assigned in iteration t. We say the pair gets *separated* in iteration t if it gets settled in iteration t and exactly one of u, v gets assigned. Thus conditioned on (u, v) not being already settled, the pair (u, v) gets settled in iteration t if $|Y_t \cap \{u, v\}| \geq 1$, and gets separated if $|Y_t \cap \{u, v\}| = 1$. Hence conditioned on being settled in iteration t, it gets separated with probability $\frac{\Pr[|Y_t \cap \{u,v\}|=1]}{\Pr[Y_t \cap \{u,v\}|\geq 1]}$. Since the time steps are independent and these probability values are the same regardless of the iteration t, it is easily seen that

$$p_{uv} = \frac{\Pr[|Y_t \cap \{u, v\}| = 1]}{\Pr[|Y_t \cap \{u, v\}| \geq 1]}.$$

Given that Y_t is distributed according to $\{\frac{x_{S,j}}{k}\}_{j \in [k], S \in \mathcal{F}_j}$, the numerator is exactly $\frac{1}{k}\sum_{j=1}^{k} \sum_{S \in \mathcal{F}_j : (u,v) \in \delta(S)} x_{S,j}$. The denominator is at least $\Pr[u \in Y_t] \geq \frac{1}{k}\sum_{j=1}^{k} \sum_{S \in \mathcal{F}_j : u \in S} x_{S,j} \geq \frac{1}{k}$ by constraint (3.1). The claim follows. □

The following theorem is immediate from Lemma 4.1. Let $\text{cost}(ALG)$ denote the cost of the partition reported by the algorithm, and let $\text{cost}(\mathcal{P})$ be the cost of the LP solution $\vec{x}$, i.e., the objective value of $\mathcal{P}$.

THEOREM 4.1. $\mathbb{E}[\text{cost}(ALG)] \leq 2 \cdot \text{cost}(\mathcal{P})$.

Proof. Lemma 4.1 implies that

$$\begin{aligned}\mathbb{E}[\text{cost}(ALG)] &= \sum_{(u,v) \in E} w(u, v) \cdot p_{u,v} \\ &\leq \sum_{(u,v) \in E} w(u, v) \sum_{j=1}^{k} \sum_{S \in \mathcal{F}_j : (u,v) \in \delta(S)} x_{S,j} \\ &= \sum_{j=1}^{k} \sum_{S \in \mathcal{F}_j} \delta(S) \cdot x_{S,j} = 2 \cdot \text{cost}(\mathcal{P}).\end{aligned}$$

□

4.3 Capacity Analysis. Let $x_{u,i}$ denote the total coverage of vertex $u \in V$ by cuts from the ith mega-bucket, formally

$$x_{u,i} \triangleq \sum_{j \in W_i} \sum_{S \in \mathcal{F}_j : u \in S} x_{S,j}.$$

It will be convenient to normalize it by the total coverage of u, formally defining

$$\tilde{x}_{u,i} \triangleq \frac{x_{u,i}}{\sum_{i'=1}^{l} x_{u,i'}} = \frac{x_{u,i}}{\sum_{j=1}^{k} \sum_{S \in \mathcal{F}_j : u \in S} x_{S,j}}.$$

The following two observations bound the probability that a vertex is assigned to a mega-bucket, and the probability that a vertex is not assigned in the first $t-1$ iterations.

OBSERVATION 1. *For every vertex $u \in V$ and mega-bucket $i = 1, \dots, \ell$,*

$$\Pr\left[m(u) = i\right] = \tilde{x}_{u,i}.$$

Proof. Fix an iteration $t \geq 1$, and condition on u being assigned in this iteration. Then

$$\begin{aligned}
&\Pr[m(u) = i \mid u \in Z_t] = \\
&= \frac{\Pr[m(u) = i \wedge u \in Z_t]}{\Pr[u \in Z_t]} \\
&= \frac{\Pr[u \notin \cup_{s=1}^{t-1} Y_s] \cdot \sum_{j \in W_i} (\frac{1}{k} \sum_{S \in \mathcal{F}_j : u \in S} x_{S,j})}{\Pr[u \notin \cup_{s=1}^{t-1} Y_s] \cdot \sum_{j=1}^{k} (\frac{1}{k} \sum_{S \in \mathcal{F}_j : u \in S} x_{S,j})} \\
&= \tilde{x}_{u,i}.
\end{aligned}$$

The observation now follows immediately by the law of total probability. □

OBSERVATION 2. *For every vertex $u \in V$ and iteration $t \geq 1$,*

$$\Pr\left[u \notin \cup_{s=1}^{t-1} Y_s\right] \leq \left(1 - \tfrac{1}{k}\right)^{t-1}.$$

Proof. Fix $u \in V$ and $s \leq t-1$, then using constraint (3.1),

$$\Pr[u \in Y_s] = \sum_{j=1}^{k} \tfrac{1}{k} \sum_{S \in \mathcal{F}_j : u \in S} x_{S,j} \geq \tfrac{1}{k}.$$

The observation now follows from the independence of the different iterations. □

4.3.1 A Martingale. We define a martingale for every mega-bucket $i = 1, \ldots, \ell$. Consider the following random variables:

$$R_{i,t} \triangleq \begin{cases} |Z_t| & \text{if } m(J_t) = i; \\ 0 & \text{if } m(J_t) \neq i; \end{cases}$$
$$N_i \triangleq \sum_{t \geq 1} R_{i,t}.$$

The random variable $R_{i,t}$ counts the number of vertices in the cut Z_t if it was added to $\mathcal{W}_i$ and 0 otherwise. N_i counts the total number of vertices in cuts added in all iterations of the algorithm to $\mathcal{W}_i$. We define the Doob martingale of N_i conditioned on the random choices the algorithm made so far:

$$M_{i,t} \triangleq \mathbb{E}\left[N_i | Y_1, \ldots, Y_t\right].$$

By definition the sequence $\{M_{i,t}\}_{t=0}^{\infty}$ is a martingale with respect to $\{Y_t\}_{t=1}^{\infty}$. Specifically, for every iteration $t \geq 1$ the following holds: $\mathbb{E}\left[M_{i,t} | Y_1, \ldots, Y_{t-1}\right] = M_{i,t-1}$.

Comment: We note that in each iteration t the algorithm makes two random choices, J_t and Y_t. Formally, in the definition of the martingale, when conditioning on the history of the random choices the algorithm made so far, we should have used (J_t, Y_t) instead of Y_t. However, for notational simplicity we only write Y_t but the reader should keep in mind that each Y_t is accompanied by the corresponding J_t.

4.3.2 Unconditioned Variance of Martingale Differences. We start by explicitly writing the change in the value of the martingale $\{M_{i,t}\}_{t=0}^{\infty}$ in every time step t.

LEMMA 4.2. *For every mega-bucket $i = 1, \ldots, \ell$ and iteration $t \geq 1$,*

$$M_{i,t} - M_{i,t-1} = R_{i,t} - \sum_{u \in Z_t} \tilde{x}_{u,i}.$$

Proof. We can write $M_{i,t}$ as

$$\begin{aligned}
M_{i,t} &\overset{\text{(i)}}{=} \sum_{s=1}^{t} R_{i,s} + \mathbb{E}\left[\sum_{s>t} R_{i,s} \middle| Y_1, \ldots, Y_t\right] \\
&= \sum_{s=1}^{t} R_{i,s} + \sum_{u \notin \cup_{s=1}^{t} Y_s} \Pr\left[m(u) = i \middle| Y_1, \ldots, Y_t\right] \\
&\overset{\text{(ii)}}{=} \sum_{s=1}^{t} R_{i,s} + \sum_{u \notin \cup_{s=1}^{t} Y_s} \tilde{x}_{u,i}.
\end{aligned}$$

Equality (i) holds since the random variables $Y_1, \ldots, Y_t$ (along with the corresponding $J_1, \ldots, J_t$) determine the value of all random variables $R_{i,1}, \ldots, R_{i,t}$. Equality (ii) follows from (the proof of) Observation 1. The lemma follows by using the above formula for t and for $t-1$, and the fact that $\cup_{s=1}^{t} Y_s \setminus \cup_{s=1}^{t-1} Y_s = Z_t$. □

Our goal is to bound the unconditioned variance of the difference of a single step of the martingale, namely $\mathrm{Var}[M_{i,t} - M_{i,t-1}]$. Since $\{M_{i,t}\}_{t=0}^{\infty}$ is a martingale, $\mathbb{E}[M_{i,t} - M_{i,t-1}] = 0$, and thus it suffices to bound the second moment of the martingale difference. Observe that $\tilde{x}_{u,i} \in [0,1]$ by its definition, and hence $|Z_t| \geq \sum_{u \in Z_t} \tilde{x}_{u,i} \geq 0$ (with probability 1). Using Lemma 4.2, it follows that $\mathrm{Var}[M_{i,t} - M_{i,t-1}] = \mathbb{E}[(M_{i,t} - M_{i,t-1})^2]$ is upper bounded by

(4.3)

$$\begin{aligned}
d_{i,t} \triangleq\ &\Pr\left[m(J_t) = i\right] \cdot \mathbb{E}\left[|Z_t|^2 \middle| m(J_t) = i\right] + \\
&\Pr\left[m(J_t) \neq i\right] \cdot \mathbb{E}\left[\left(\sum_{u \in Z_t} \tilde{x}_{u,i}\right)^2 \middle| m(J_t) \neq i\right].
\end{aligned}$$

The above is correct since one can examine the two different cases. If $m(J_t) = i$ then by definition $R_{i,t} = |Z_t|$ and we get that $M_{i,t} - M_{i,t-1} = |Z_t| - \sum_{u \in Z_t} \tilde{x}_{u,i} \geq 0$, and therefore $(M_{i,t} - M_{i,t-1})^2 \leq |Z_t|^2$. Otherwise $m(J_t) \neq i$ and in this case $R_{i,t} = 0$ and $(M_{i,t} - M_{i,t-1})^2 = \left(\sum_{u \in Z_t} \tilde{x}_{u,i}\right)^2$. The next lemma crucially provides an upper bound on $d_{i,t}$.

LEMMA 4.3. *For every $1 \le i \le \ell$ and every $t \ge 1$,*

$$d_{i,t} \le \frac{2k_i}{k}\left(1-\frac{1}{k}\right)^{t-1} 2^{-(i-1)} n_1^2 .$$

Proof. We start by bounding the first term of $d_{i,t}$.

$$\begin{aligned}
&\Pr\left[m(J_t)=i\right]\cdot \mathbb{E}\left[|Z_t|^2 \middle| m(J_t)=i\right] = \\
&= \frac{k_i}{k}\sum_{u,v\in V}\Pr\left[u,v\in Z_t \middle| m(J_t)=i\right] \\
&= \frac{k_i}{k}\sum_{u,v\in V}\Pr\left[u,v\notin \cup_{s=1}^{t-1}Y_s \wedge u,v\in Y_t \middle| m(J_t)=i\right] \\
&\overset{\text{(i)}}{=} \frac{k_i}{k}\sum_{u,v\in V}\Pr\left[u,v\notin \cup_{s=1}^{t-1}Y_s\right]\cdot\Pr\left[u,v\in Y_t \middle| m(J_t)=i\right] \\
&\overset{\text{(ii)}}{\le} \frac{k_i}{k}\left(1-\frac{1}{k}\right)^{t-1}\cdot\sum_{u,v\in V}\Pr\left[u,v\in Y_t \middle| m(J_t)=i\right] \\
&= \frac{k_i}{k}\left(1-\frac{1}{k}\right)^{t-1}\cdot\mathbb{E}\left[|Y_t|^2 \middle| m(J_t)=i\right]
\end{aligned}$$

(4.4)
$$\overset{\text{(iii)}}{\le} \frac{k_i}{k}\left(1-\frac{1}{k}\right)^{t-1}\cdot 2^{-2(i-1)}n_1^2 .$$

Equality (i) is derived from the fact that all iterations of the algorithm are independent. Inequality (ii) follows from Observation 2 and the fact that $\Pr\left[u,v\notin\cup_{s=1}^{t-1}Y_s\right] \le \Pr\left[u\notin\cup_{s=1}^{t-1}Y_s\right]$. Finally, inequality (iii) holds since Y_t is chosen from mega-bucket i and must be of size at most $2^{-(i-1)}n_1$.

We now upper bound the second term of $d_{i,t}$.

$$\begin{aligned}
&\Pr\left[m(J_t)\neq i\right]\cdot\mathbb{E}\left[\left(\sum_{u\in Z_t}\tilde{x}_{u,i}\right)^2 \middle| m(J_t)\neq i\right] = \\
&= \sum_{r\neq i}\Pr\left[m(J_t)=r\right]\cdot\mathbb{E}\left[\left(\sum_{u\in Z_t}\tilde{x}_{u,i}\right)^2 \middle| m(J_t)=r\right] \\
&= \sum_{r\neq i}\frac{k_r}{k}\sum_{u,v\in V}\tilde{x}_{u,i}\tilde{x}_{v,i}\cdot\Pr\left[u,v\in Z_t \middle| m(J_t)=r\right] \\
&\overset{\text{(iv)}}{=} \sum_{r\neq i}\frac{k_r}{k}\sum_{u,v\in V}\Bigg[\tilde{x}_{u,i}\tilde{x}_{v,i}\cdot\Pr\left[u,v\notin\cup_{s=1}^{t-1}Y_s\right]\cdot \\
&\qquad\qquad \Pr\left[u,v\in Y_t \middle| m(J_t)=r\right]\Bigg] \\
&\overset{\text{(v)}}{\le} \left(1-\frac{1}{k}\right)^{t-1}\sum_{r\neq i}\frac{k_r}{k}\sum_{u,v\in V}\Bigg[\tilde{x}_{u,i}\tilde{x}_{v,i}\cdot \\
&\qquad \Pr\left[u\in Y_t \middle| m(J_t)=r\right]\Pr\left[v\in Y_t \middle| u\in Y_t, m(J_t)=r\right]\Bigg] \\
&\overset{\text{(vi)}}{=} \frac{1}{k}\left(1-\frac{1}{k}\right)^{t-1}\sum_{r\neq i}\sum_{u,v\in V}\Bigg[\tilde{x}_{u,i}\tilde{x}_{v,i}x_{u,r}\cdot \\
&\qquad \Pr\left[v\in Y_t \middle| u\in Y_t, m(J_t)=r\right]\Bigg] \\
&\overset{\text{(vii)}}{\le} \frac{1}{k}\left(1-\frac{1}{k}\right)^{t-1}\sum_{r\neq i}2^{-(r-1)}n_1\sum_{u\in V}\tilde{x}_{u,i}x_{u,r}
\end{aligned}$$

(4.5)
$$\overset{\text{(viii)}}{\le} \frac{k_i}{k}\left(1-\frac{1}{k}\right)^{t-1}2^{-(i-1)}n_1^2\sum_{r\neq i}2^{-(r-1)} .$$

As before, equality (iv) follows from the independence of the algorithm's iterations, and inequality (v) follows from Observation 2 and the fact that $\Pr\left[u,v\notin\cup_{s=1}^{t-1}Y_s\right] \le \Pr\left[u\notin\cup_{s=1}^{t-1}Y_s\right]$. Note that equality (vi) holds since $\Pr[u\in Y_t \mid m(J_t)=r] = \frac{1}{k_r}\sum_{j\in W_r}\sum_{S\in\mathcal{F}_j:u\in S}x_{S,j} = \frac{1}{k_r}x_{u,r}$. Inequality (vii) is derived from $\tilde{x}_{v,i}\in[0,1]$ and an upper bound on the size of a set Y_t chosen from mega-bucket W_r, namely

$$\sum_{v\in V}\tilde{x}_{v,i}\cdot\Pr\left[v\in Y_t \middle| u\in Y_t, m(J_t)=r\right] \le$$
$$\mathbb{E}\left[|Y_t| \middle| u\in Y_t, m(J_t)=r\right] \le 2^{-(r-1)}n_1 .$$

Inequality (viii) is derived by considerable rearranging and similar facts like $\tilde{x}_{u,r}\in[0,1]$ and constraint (3.2), namely,

$$\begin{aligned}
\sum_{u\in V}\tilde{x}_{u,i}x_{u,r} &= \sum_{u\in V}x_{u,i}\tilde{x}_{u,r} \\
&\le \sum_{u\in V}x_{u,i} \\
&= \sum_{j\in W_i}\sum_{S\in\mathcal{F}_j}\sum_{u\in S}x_{S,j} \\
&= \sum_{j\in W_i}\sum_{S\in\mathcal{F}_j}\left(|S|\cdot x_{S,j}\right) \\
&\le k_i\cdot 2^{-(i-1)}n_1 .
\end{aligned}$$

Combining the upper bounds (4.4) and (4.5) on $d_{i,t}$ we conclude that

$$\begin{aligned}
d_{i,t} &\le \frac{k_i}{k}\left(1-\frac{1}{k}\right)^{t-1}\cdot 2^{-2(i-1)}n_1^2 + \\
&\quad \frac{k_i}{k}\left(1-\frac{1}{k}\right)^{t-1}\cdot 2^{-(i-1)}n_1^2\cdot\sum_{r\neq i}2^{-(r-1)} \\
&= \frac{k_i}{k}\left(1-\frac{1}{k}\right)^{t-1}2^{-(i-1)}n_1^2\left(2^{-(i-1)}+\sum_{r\neq i}2^{-(r-1)}\right) \\
&\le \frac{2k_i}{k}\left(1-\frac{1}{k}\right)^{t-1}2^{-(i-1)}n_1^2 .
\end{aligned}$$

□

4.3.3 Sums of Conditioned Variances of Martingale Differences.

We proceed to bound the sum, over all iterations of the algorithm, of the *conditioned* variances of the martingale differences for a fixed mega-bucket W_i.

LEMMA 4.4. *For every $i = 1, \ldots, \ell$ and $\alpha_i > 1$, let A be the event that $\sum_{t \geq 1} \mathrm{Var}[M_{i,t} - M_{i-t-1} | Y_1, \ldots, Y_{t-1}] \geq \alpha_i \cdot k_i \cdot 2^{-(i-2)} n_1^2$. Then,*

$$\Pr[A] \leq \frac{1}{\alpha_i} .$$

Proof. Similarly to (4.3), we can bound

$$\mathrm{Var}[M_{i,t} - M_{i,t-1} \Big| Y_1, \ldots, Y_{t-1}]$$

from above by

$$D_{i,t} \triangleq \Bigg(\Pr[m(J_t) = i] \cdot \mathbb{E}\left[|Z_t|^2 \Big| m(J_t) = i, \{Y_r\}_{1 \leq r \leq t-1} \right] \Bigg) + \Bigg(\Pr[m(J_t) \neq i] \cdot \mathbb{E}\left[\left(\sum_{u \in Z_t} \tilde{x}_{u,i} \right)^2 \Bigg| m(J_t) \neq i, \{Y_r\}_{1 \leq r \leq t-1} \right] \Bigg) .$$

Notice that $D_{i,t}$, in contrast to $d_{i,t}$, is a random variable depending on $Y_1, \ldots, Y_{t-1}$. By the law of total probability, $\mathbb{E}[D_{i,t}] = d_{i,t}$, and thus, using linearity of expectation and Lemma 4.3,

$$\begin{aligned} \mathbb{E}\Big[\sum_{t \geq 1} D_{i,t} \Big] &= \sum_{t \geq 1} d_{i,t} \\ &\leq \sum_{t \geq 1} \frac{2k_i}{k} \left(1 - \frac{1}{k} \right)^{t-1} 2^{-(i-1)} n_1^2 \\ &= k_i 2^{-(i-2)} n_1^2 . \end{aligned}$$

Applying Markov's inequality, we obtain a bound on the upper tail of $\sum_{t \geq 1} D_{i,t}$, and the lemma follows. □

4.3.4 Martingale Concentration via Stopping Times.

For every mega-bucket W_i, $1 \leq i \leq \ell$, we are given a martingale $\{M_{i,t}\}_{t=0}^{\infty}$ with respect to $\{Y_t\}_{t=1}^{\infty}$. Additionally, we know that except with a small probability (with the appropriate choice of α_i), Lemma 4.4 implies that the sum of conditioned variances of the martingale differences is small. How can one use this along with the martingale to obtain a good concentration? Obviously one can condition on the event that the sum of variances is small for all mega-buckets (i.e., the algorithm repeats until this event happens). However, once conditioned on this event the sequence $\{M_{i,t}\}_{t=0}^{\infty}$ is not guaranteed to be a martingale anymore. To overcome this, we use a martingale concentration bound that depends on stopping times, as stated below.

THEOREM 4.2. (FREEDMAN'S INEQUALITY [FRE75]) *Let $\{X_n\}_{n=1}^{\infty}$ be a martingale difference with respect to $\{Y_n\}_{n=1}^{\infty}$, namely, $\mathbb{E}[X_n | Y_1, \ldots, Y_{n-1}] = 0$ for every $n \geq 1$. Assume in addition that $|X_n| \leq 1$ (with probability 1). Let $S_n \triangleq \sum_{i=1}^{n} X_i$ and $T_n \triangleq \sum_{i=1}^{n} \mathrm{Var}[X_i | Y_1, \ldots, Y_{i-1}]$. Then*

$$\Pr[S_n \geq \gamma \textit{ and } T_n \leq \delta \textit{ for some } n] \leq e^{-\frac{\gamma^2}{2(\gamma+\delta)}} ,$$

for every $\gamma, \delta > 0$.

The following theorem shows that the algorithm outputs a partition that deviates from the desired capacities by at most a constant factor. Its proof uses Freedman's inequality.

THEOREM 4.3. *If the instance is b-nice for a large enough constant b, then with a constant probability, the partition $\cup_{i=1}^{\ell} \mathcal{W}_i$ reported by the algorithm deviates from the capacity bounds by at most a constant factor.*

Proof. We will use Freedman's inequality for each of the non-empty megabuckets. Recall that since the instance is *b-nice*, there is a sequence of integers $1 = p_1, \ldots, p_l$ such that $k_{p_i} 2^{-(p_i - 1)} n_1 \geq b \cdot k_{p_{i-1}} 2^{-(p_{i-1}-1)} n_1$ for all $i \geq 2$, and these are the only non-empty mega-buckets. It follows that

$$k_{p_i} 2^{-(p_i - 1)} \geq b^{i-1}, \qquad \forall i > 1.$$

We will show that with a constant probability, the multiplicative deviations of all the martingales corresponding to non-empty mega-buckets are upper bounded by a constant.

Given a fixed mega-bucket W_{p_i} we choose the following parameters for Freedman's inequality, where $C, \alpha > 1$ are absolute constants to be determined later:

$$\begin{aligned} X_{i,t} &\triangleq \frac{M_{p_i,t} - M_{p_i,t-1}}{n_1} \\ \gamma_i &\triangleq C \cdot k_{p_i} \cdot 2^{-(p_i - 1)} \\ \delta_i &\triangleq C \cdot \alpha^i \cdot k_{p_i} \cdot 2^{-(p_i - 1)} . \end{aligned}$$

Let us verify that all the conditions of Freedman's inequality (Theorem 4.2) are satisfied. First, note that $\mathbb{E}[X_{i,t} | Y_1, \ldots, Y_{t-1}] = 0$ since $\{M_{p_i,t}\}_{t=0}^{\infty}$ is a martingale with respect to $\{Y_t\}_{t=1}^{\infty}$. Second, by Lemma 4.2 and the fact that $0 \leq \tilde{x}_{u,p_i} \leq 1$,

$$|M_{p_i,t} - M_{p_i,t-1}| \leq \begin{cases} |Z_t| & \text{if } m(J_t) = p_i; \\ \sum_{u \in Z_t} \tilde{x}_{u,p_i} & \text{if } m(J_t) \neq p_i. \end{cases}$$

In the first case, $|Z_t| \leq 2^{-(p_i-1)} n_1 \leq n_1$. In the second case, $\sum_{u \in Z_t} \tilde{x}_{u,p_i} \leq |Z_t| \leq n_1$, since $\tilde{x}_{u,p_i} \leq 1$ by definition and n_1 is an upper bound on the size of *all* buckets. Therefore, it is always the case that $|X_{i,t}| = |M_{i,t} - M_{i,t-1}|/n_1 \leq 1$. Hence, all the conditions of Freedman's inequality are satisfied.

By Lemma 4.4, we can bound the sum of conditioned variances of $\{X_{i,t}\}_{t \geq 1}$ as follows:

$$\Pr\Big[\sum_{t\geq 1} \mathrm{Var}[X_{i,t} \mid Y_1, \ldots, Y_{t-1}] \geq \delta_i\Big] \leq \frac{2}{C\alpha^i} .$$

Let B_i be the event in the above expression. Applying Freedman's inequality to the martingale difference sequence $\{X_{i,t}\}_{t\geq 1}$ gives,

$$\begin{aligned}\Pr\Big[\sum_{t\geq 1} X_{i,t} \geq \gamma_i \ \wedge \ \overline{B}_i\Big] &\leq \exp\Big(-\frac{\gamma_i^2}{2(\gamma_i+\delta_i)}\Big)\\ &= \exp\Big(-\frac{C \cdot k_{p_i} 2^{-(p_i-1)}}{2(1+\alpha^i)}\Big)\\ &\leq \exp\Big(-\frac{Cb^{i-1}}{2\alpha^i}\Big) ,\end{aligned}$$

and together we have

$$\begin{aligned}\Pr\Big[\sum_{t\geq 1} X_{i,t} \geq \gamma_i\Big] &\leq \Pr[B_i] + \Pr\Big[\sum_{t\geq 1} X_{i,t} \geq \gamma_i \ \wedge \ \overline{B}_i\Big]\\ &\leq \frac{2}{C\alpha^i} + \exp\Big(-\frac{Cb^i}{2\alpha^i}\Big) .\end{aligned}$$

Let us set $\alpha = 2$, $b = 8$ and $C = 4$. Plugging these in and summing up over all i, we get:

$$\begin{aligned}&\Pr\Big[\exists 1 \leq i \leq \ell, \sum_{t\geq 1} X_{i,t} \geq \gamma_i\Big] \leq\\ &\qquad \leq \sum_{i=1}^{l} \Pr\Big[\sum_{t\geq 1} X_{i,t} \geq \gamma_i\Big]\\ &\qquad \leq \sum_{i=1}^{l} \frac{1}{2 \cdot 2^i} + \sum_{i=1}^{l} \exp\Big(-\frac{2 \cdot 8^{(i-1)}}{2^i}\Big)\\ &\qquad < 0.9 .\end{aligned}$$

It follows that with a probability of at least a constant, all mega-buckets have their capacity violated by at most a constant factor of C. Since each cut Z_t added to mega-bucket p_i has size at most $2^{-(p_i-1)} n_1$, it is easy to check that the repeated merging of small pieces in steps 10-12 of the algorithm produces k_{p_i} pieces of size at most $(C+1) \cdot 2^{-(p_i-1)} n_1$. Thus all capacities are violated by at most a factor of $C + 1 = 5$. □

Proof. [of Theorem 1.1] Follows from Theorem 2.1, Theorem 3.1 (and its subsequent discussion in Section 3), Theorem 4.1, and from Theorem 4.3. □

5 Solving the Configuration LP

In this section we prove Theorem 3.1. In principle, all we need is an approximate separation oracle for the dual of $(\mathcal{P})$. However, since $(\mathcal{P})$ is a mixed packing and covering program, the dual's objective function contains negative terms, and it is a little more tricky to approximate its overall value (rather than approximating the positive and negative terms separately). We follow the method of Chakrabarty and Swamy [CS11, Lemma 3.3 in the arxiv version], where the dual LP is "massaged", scaling some variables and constraints by small fudge factors $a, b \geq 1$. Section 5.2 proves Theorem 3.1 by describing such an algorithm and its analysis.[2] Section 5.3 gives an alternative proof of the same theorem by using (with some needed modifications) Young's iterative method [You01] for quickly solving fast packing and covering problems. We note that a black-box application of Young's technique does not provide the guarantee we need, and a different rescaling of the objective packing constraint (once the problem is reduced to a feasibility question) is needed.

5.1 Weighted Unbalanced Cut.

We require an algorithm for the following problem:

DEFINITION 5.1. *An instance of the* WEIGHTED UNBALANCED CUT *problem is given by (G, r, y, Y), where $G = (V, E)$ is an undirected graph with non-negative weights on the edges, a capacity parameter r, a non-negative weight function on the vertices $y : V \to \mathbb{R}_+$ and a weight parameter Y. The goal is to find a cut $S \subseteq V$ subject to $|S| \leq r$ and $y(S) \geq Y$ that minimizes $\delta(S)$.*

The following lemma is implicit in the work of Räcke [RÖ8].

LEMMA 5.1. *There is an efficient algorithm that given an instance (G, r, y, Y) of* WEIGHTED UNBALANCED CUT *and any absolute constant $\alpha > 1$, finds a cut S such that $|S| \leq r$, $y(S) \geq Y/\alpha$ and $\delta(S) \leq L \cdot \delta(S^*)$ for some optimal solution S^*, where $L = O(\log n)$.*

5.2 Solving the LP via a Separation Oracle for the Dual.

The dual of $(\mathcal{P})$, our configuration LP, is given by the following LP:

$$\begin{aligned}(\mathcal{D}) \quad \max \quad & \sum_{v\in V} y_v - \sum_{j=1}^{k} z_j\\ \text{s.t.} \quad & \sum_{v\in S} y_v - z_j \leq \delta(S) \qquad \forall j = 1, \ldots, k, \ \forall S \in \mathcal{F}_j\\ & y_v \geq 0, \ z_j \geq 0 \qquad \forall v \in V, \ \forall j = 1, \ldots, k\end{aligned}$$

[2]The proof here was suggested to us by an anonymous reviewer. Our original proof follows the same outline, and is similar except that the fudge factors manipulation is slightly more specialized to our context.

Given values $(y_v : v \in V)$ (or similarly $y : V \to \mathbb{R}_+$), define for $S \subset V$ its aggregate value $y(S) \triangleq \sum_{v \in S} y_v$. Now, for $d \in \mathbb{R}$ and $a, b \geq 1$, define the following polytope (feasibility LP):

$$\mathcal{D}'(d,a,b) \triangleq$$

$$\Big\{ \; y(V) - a \sum_{j=1}^{k} z_j \geq d,$$

$$y(S) - \tfrac{1}{2b}\, \delta(S) \leq z_j \quad \forall j = 1, \ldots, k,\ \forall S \in \mathcal{F}_j, \tag{5.6}$$

$$y_v \geq 0,\ z_j \geq 0, \qquad \forall v \in V,\ \forall j = 1, \ldots, k \Big\}.$$

Observe that $\mathrm{OPT}(\mathcal{D})$, the optimum (objective value) of $(\mathcal{D})$, equals the maximum d such that $\mathcal{D}'(d,1,1)$ is non-empty. Througout, we set $a \triangleq (1+\varepsilon)^2$ and $b \triangleq aL$ (where ε is from Theorem 3.1 and L is from Lemma 5.1).

LEMMA 5.2. *There is a polynomial-time procedure that given d and vectors $\vec{y}, \vec{z}$, either reports a constraint of $\mathcal{D}'(d,a,b)$ violated by $(\vec{y}, \vec{z})$, or determines that $(\vec{y}, a\vec{z}) \in \mathcal{D}'(d,1,1)$.*

Proof. Given d and vectors $\vec{y}, \vec{z}$, first check the first and last sets of constraints used to define $\mathcal{D}'(d,a,b)$. If any of these constraints is violated, we can report it; so assume henceforth they are satisfied. Next, to check constraints (5.6), do the following for each $j = 1, \ldots, k$. Start by exhaustively checking the constraint for all singletons, i.e, $S \subset V$ with $|S| = 1$. Again, if any of these constraints is violated, we can report it; so assume henceforth they are satisfied. It follows that $y(V) \leq nz_j + 2w(E)$. Whenever $y(S) \leq z_j$, the corresponding constraint is surely satisfied; thus, we only need to check subsets S where $y(S) \in [z_j, nz_j + 2w(E)]$. Apply Lemma 5.1 with $\alpha = 1 + \varepsilon$ and every $Y \in [\frac{1}{1+\varepsilon}\, z_j, nz_j + 2w(E)]$ that is a power of $1+\varepsilon$. If, for any of those Y values, the obtained set $T \subset V$ satisfies $y(T) - \frac{1}{2b}\,\delta(T) > z_j$, then we have a constraint of $\mathcal{D}'(p,a,b)$ violated by $(\vec{y}, \vec{z})$, and we can report it. Otherwise, we know that for every subset S with $y(S) \in [Y, (1+\varepsilon)Y]$,

$$\begin{aligned} y(S) - \tfrac{1}{2}\,\delta(S) &\leq \alpha(1+\varepsilon) y(T) - \tfrac{1}{2L}\,\delta(T) \\ &= [y(T) - \tfrac{1}{2b}\,\delta(T)] \\ &\leq a z_j \,. \end{aligned}$$

If we have not found a violated constraint for any of the Y values, then we know that all $S \in \mathcal{F}_j$ satisfy $y(S) - \frac{1}{2b}\,\delta(S) \leq a z_j$, and since we have checked all other constraints explicitly, we can determine that $(\vec{y}, a\vec{z}) \in \mathcal{D}'(p,1,1)$.

Finally, this entire procedure runs in polynomial time because the number of Y values that are examined for each $j = 1, \ldots, k$ is at most $O(\frac{1}{\varepsilon}\log(n + w(E)/z_j))$. □

We now continue the proof of Theorem 3.1. Using the Ellipsoid algorithm and Lemma 5.2 as an approximate separation oracle, we can solve the following in polynomial time: Given $p \in \mathbb{R}$, either find some $(\vec{y}, a\vec{z}) \in \mathcal{D}'(d,1,1)$ or determine that $\mathcal{D}'(d,a,b) = \emptyset$. Now perform a binary search on d until reaching additive precision $\gamma > 0$ (set to be exponentially small in the input size), and let d^* be the largest d for which the former outcome is obtained. In particular, we have found some $(\vec{y}^*, a\vec{z}^*) \in \mathcal{D}'(d^*,1,1)$, implying that $\mathcal{D}$ has a feasible solution $(\vec{y}^*, a\vec{z}^*)$ and optimal value $\mathrm{OPT}(\mathcal{D}) \geq \sum_v y_v^* - a\sum_j z_j^* \geq d^*$.

For value $d = d^* + \gamma$, the Ellipsoid algorithm produced a polynomial-size collection of constraints that certifies the emptiness of $\mathcal{D}'(d,a,b)$. It follows that that the "compact-dual", which seeks to maximize $y(V) - a\sum_j z_j$ subject to that polynomial-size subset of (5.6) and nonnegativity constraints, has value smaller than $d = d^* + \gamma$. Since γ is exponentially small in the input size (recall $(\mathcal{P})$ is succinctly represent using G and $n_1, \ldots, n_k$), it is also exponentially small relative to the size of our compact-dual, the said maximum (of the compact-dual) must be at most d^*. The LP dual of the compact-dual is similar to our original primal $(\mathcal{P})$, except that (i) it has only polynomially many variables $x_{S,j}$; (ii) the objective function coefficients are $\frac{1}{2b}\,\delta(S)$; and (iii) the righthand side of (3.2) is a. This LP has polynomial size and by duality its optimal value is at most d^*. Now solve this LP and interpret the solution found as a solution to $(\mathcal{P})$ by setting the remaining $x_{S,j}$ variables to zero (implicitly). Overall, our algorithm runs in polynomial time and computes an approximate solution for $(\mathcal{P})$ in the sense that its value is at most $bd^* \leq aL\,\mathrm{OPT}(\mathcal{D}) \leq O(\log n)\,\mathrm{OPT}(\mathcal{P})$ and constraints (3.2) are violated by at most factor $a = (1+\varepsilon)^2$. This completes the proof of Theorem 3.1.

5.3 Solving the LP by Young's Iterative Method.

For simplicity, let us reduce $(\mathcal{P})$ to a feasibility problem. Consider the following polytope $\mathcal{Q}$ parameterized by M:

$$\mathcal{Q}(M) = \Big\{ \sum_{j=1}^{k} \sum_{S \in \mathcal{F}_j : u \in S} x_{S,j} \geq 1 \qquad \forall u \in V\,, \tag{5.7}$$

$$\sum_{S \in \mathcal{F}_j} x_{S,j} \leq 1 \qquad \forall j = 1, \ldots, k\,, \tag{5.8}$$

$$\frac{1}{M \cdot L} \sum_{j=1}^{k} \sum_{S \in \mathcal{F}_j} \delta(S) \cdot x_{S,j} \leq 1 \tag{5.9}$$

$$x_{S,j} \geq 0 \qquad \forall j = 1, \ldots, k, \forall S \in \mathcal{F}_j \;\Big\}$$

M is our guess for the value of an optimal solution to the NON-UNIFORM GRAPH PARTITIONING instance we have (which can be guessed in polynomial time up to a constant be rescaling of the edge weights and then applying binary search). Denote by C the matrix of the covering constraints of $\mathcal{Q}$, namely (5.7), and by P the matrix of the packing constraints of $\mathcal{Q}$, namely (5.8) and (5.9). It is important to note that we scaled the objective constraint (5.9) by an additional factor of L as to compensate for the loss in the solution of WEIGHTED UNBALANCED CUT.

Following the footsteps of Young [You01], we define the following smooth approximations of the max and min functions for any vector $\vec{x} = (x_{S,j})_{1\le j\le k, S\in\mathcal{F}_j}$ indexed by $j = 1,\dots,k$ and $S \in \mathcal{F}_j$:

$$lmax(P\cdot\vec{x}) \triangleq \ln\left(R(\vec{x}) + \sum_{j=1}^{k} z(\vec{x},j)\right)$$
$$lmin(C\cdot\vec{x}) \triangleq -\ln\left(\sum_{u\in V} y(\vec{x},u)\right) ,$$

where,

$$R(\vec{x}) \triangleq \exp\left(\frac{N}{M\cdot L}\sum_{j=1}^{k}\sum_{S\in\mathcal{F}_j}\delta(S)\cdot x_{S,j}\right)$$
$$z(\vec{x},j) \triangleq \exp\left(N\sum_{S\in\mathcal{F}_j} x_{S,j}\right)$$
$$y(\vec{x},u) \triangleq \exp\left(-N\sum_{j=1}^{k}\sum_{S\in\mathcal{F}_j:u\in S} x_{S,j}\right) .$$

N is a scaling parameter to be chosen later and L is the approximation factor from Lemma 5.1. Note that $R(\vec{x})$ corresponds to the packing constraint (5.9), $z(\vec{x},j)$ corresponds to the packing constraint (5.8), and $y(\vec{x},u)$ corresponds to the covering constraint (5.7).

The reader should keep in mind while reading the algorithm and its proof that:

(5.10)
$$lmax(P\cdot\vec{x}) \ge N\cdot\max\left\{\frac{1}{M\cdot L}\sum_{j=1}^{k}\sum_{S\in\mathcal{F}_j}\delta(S)\cdot x_{S,j}\ ,\ \sum_{S\in\mathcal{F}_1} x_{S,1}\ ,\ \dots\ ,\ \sum_{S\in\mathcal{F}_k} x_{S,k}\right\}$$

(5.11)
$$lmin(C\cdot\vec{x}) \le N\cdot\min_{u\in V}\left\{\sum_{j=1}^{k}\sum_{S\in\mathcal{F}_j:u\in S} x_{S,j}\right\} .$$

The above follows immediately from the definition of $lmax$ and $lmin$. Intuitively, as the free coefficient of all constraints in $\mathcal{Q}(M)$ is 1, $lmax(P\cdot\vec{x})$ is at most N times the value of the worst packing constraint (the one closest to being violated). On the other hand, $lmin(C\cdot\vec{x})$ is at least N times the value of the worst covering constraint (the one furthest from being satisfied).

Let us describe now the algorithm of finding a point in $\mathcal{Q}(M)$, where $\beta > 1$ is an arbitrary absolute constant and $\gamma > 0$ is a small step size to be determined later. The algorithm starts with $\vec{x} = 0$ and increases it as long as $\vec{x}$ violates at least one of the covering constraints.

```
1:  $\vec{x} \leftarrow 0$.
2:  while $\exists u\in V$ s.t. $\sum_{j=1}^{k}\sum_{S\in\mathcal{F}_j:u\in S} x_{S,j} < 1$ do
3:      $r \leftarrow 0$.
4:      $Y \leftarrow \sum_{u\in V} y(\vec{x},u)$.
5:      $\forall 1\le j\le k$ use Lemma 5.1 to approximately solve
          $(G, n_j, \{y(\vec{x},u)\}_{u\in V}, Y)$ and obtain $S_{j,r}$.
6:      while
          $\min_{1\le j\le k}\left\{\frac{\sum_{u\in V} y(\vec{x},u)}{R(\vec{x})+\sum_{j'=1}^{k} z(\vec{x},j')}\cdot\frac{\frac{1}{M\cdot L}R(\vec{x})\delta(S_{j,r})+z(\vec{x},j)}{\sum_{u\in S_{j,r}} y(\vec{x},u)}\right\}>\alpha\beta$
        do
7:          $Y \leftarrow Y/\beta$.
8:          $r \leftarrow r+1$.
9:      Let $j$ be the index for which
          $\frac{\sum_{u\in V} y(\vec{x},u)}{R(\vec{x})+\sum_{j'=1}^{k} z(\vec{x},j')}\cdot\frac{\frac{1}{M\cdot L}R(\vec{x})\delta(S_{j,r})+z(\vec{x},j)}{\sum_{u\in S_{j,r}} y(\vec{x},u)}\le\alpha\beta$ .
10:     $\vec{x} \leftarrow \vec{x} + \gamma\cdot\mathbf{1}_{S_{j,r},j}$.
11: Return $\vec{x}$.
```

LEMMA 5.3. *If M is at least the value of an optimal solution to* NON-UNIFORM GRAPH PARTITIONING, *then for every iteration of the big while loop (step (2)) with $\vec{x}$, there exists a j and an r such that $S_{j,r}$ satisfies:*

$$\frac{\sum_{u\in V} y(\vec{x},u)}{R(\vec{x})+\sum_{j'=1}^{k} z(\vec{x},j')}\cdot\frac{\frac{1}{M\cdot L}R(\vec{x})\delta(S_j)+z(\vec{x},j)}{\sum_{u\in S_j} y(\vec{x},u)}\le\alpha\beta\ .$$

Proof. Consider the following problem:

$$(*)\qquad \min_{1\le j\le k}\min_{S\in\mathcal{F}_j}\left\{\frac{\sum_{u\in V} y(\vec{x},u)}{R(\vec{x})+\sum_{j'=1}^{k} z(\vec{x},j')}\cdot\frac{\frac{1}{M}R(\vec{x})\delta(S)+z(\vec{x},j)}{\sum_{u\in S} y(\vec{x},u)}\right\} ,$$

and let j^* and S^* be an optimal solution to it. It it important to note that this optimization problem is *different* from what appears in the theorem, as the coefficient of $\delta(S)$ is $\frac{1}{M}R(\vec{x})$ and not $\frac{1}{M\cdot L}R(\vec{x})$. Our proof has two steps. In the first, we prove that $(*)$ has value at most 1, namely that plugging j^* and S^* into $(*)$ results in value which is at most 1. In the second step we show that the algorithm, in the worst case, finds an approximate solution to $(*)$, thus proving the lemma.

Let us start with the first step. Define the following two vectors $\vec{z}$ and $\vec{y}$ indexed by $1 \le j \le k$ and $S \in \mathcal{F}_j$:

$$z_{S,j} \triangleq \frac{\frac{1}{M} R(\vec{x})\delta(S) + z(\vec{x}, j)}{R(\vec{x}) + \sum_{j'=1}^{k} z(\vec{x}, j')}$$
$$y_{S,j} \triangleq \frac{\sum_{u \in S} y(\vec{x}, u)}{\sum_{u \in V} y(\vec{x}, u)} .$$

Again, it is important to note that in the definition of $z_{S,j}$ to coefficient of $\delta(S)$ is only $\frac{1}{M} R(\vec{x})$. Since M is at least the value of an optimal solution to NON-UNIFORM GRAPH PARTITIONING, denoting by $\vec{x}^*$ such an integral optimal solution implies that:

$$\sum_{j=1}^{k} \sum_{S \in \mathcal{F}_j} \delta(S) \cdot x^*_{S,j} \le M .$$

Consider $\vec{z} \cdot \vec{x}^*$:

$$\vec{z} \cdot \vec{x}^* = \sum_{j=1}^{k} \sum_{S \in \mathcal{F}_j} \frac{\frac{1}{M} R(\vec{x})\delta(S) \cdot x^*_{S,j} + z(\vec{x}, j) \cdot x^*_{S,j}}{R(\vec{x}) + \sum_{j'=1}^{k} z(\vec{x}, j')}$$
$$\le \frac{R(\vec{x}) + \sum_{j=1}^{k} z(\vec{x}, j)}{R(\vec{x}) + \sum_{j'=1}^{k} z(\vec{x}, j')} = 1 .$$

The inequality is derived from the above assumption on M and the fact that $\vec{x}^*$ is feasible, and in particular $\sum_{S \in \mathcal{F}_j} x^*_{S,j} \le 1$ for every $1 \le j \le k$. Consider now $\vec{y} \cdot \vec{x}^*$:

$$\vec{y} \cdot \vec{x}^* = \sum_{j=1}^{k} \sum_{S \in \mathcal{F}_j} \frac{x^*_{S,j} \cdot \sum_{u \in S} y(\vec{x}, u)}{\sum_{u \in V} y(\vec{x}, u)}$$
$$= \frac{\sum_{u \in V} y(\vec{x}, u) \cdot \sum_{j=1}^{k} \sum_{S \in \mathcal{F}_j : u \in S} x^*_{S,j}}{\sum_{u \in V} y(\vec{x}, u)}$$
$$\ge \frac{\sum_{u \in V} y(\vec{x}, u)}{\sum_{u \in V} y(\vec{x}, u)} = 1 .$$

The second equality is just a change in the summation order, and the inequality is derived from the fact that x^* is feasible, and in particular $\sum_{j=1}^{k} \sum_{S \in \mathcal{F}_j : u \in S} x^*_{S,j} \ge 1$ for every $u \in V$.

We proved that $\vec{z} \cdot \vec{x}^* \le 1 \le \vec{y} \cdot \vec{x}^*$. Since all the vectors $\vec{x}^*$, $\vec{z}$ and $\vec{y}$ are in the non-negative orthant, there must be some coordinate that corresponds to some j and $S \in \mathcal{F}_j$ in which $z_{S,j} \le y_{S,j}$. This implies that $(*) \le 1$, concluding the first step of the proof.

Fix an iteration of the big while loop (step (2)) and consider the first time that $Y < \sum_{u \in S^* y(\vec{x}, u)}$, which implies that $Y \ge \sum_{u \in S^* y(\vec{x}, u)} / \beta$ by the definition of the algorithm. When this happens, denote by $\tilde{S}$ the cut the algorithm obtains when it uses Lemma 5.1 to approximately solve the instance $(G, n_{j^*}, \{y(\vec{x}, u)\}_{u \in V}, Y)$ of WEIGHTED UNBALANCED CUT. Lemma 5.1 implies that

$$|\tilde{S}| \le n_{j^*} \text{ and } \sum_{u \in \tilde{S}} y(\vec{x}, u) \ge Y/\alpha \ge \sum_{u \in S^*} y(\vec{x}, u)/(\alpha\beta) ,$$

and that $\delta(\tilde{S}) \le L \cdot \delta(S^*)$. Therefore, using all these one can bound:

$$\frac{\sum_{u \in V} y(\vec{x}, u)}{R(\vec{x}) + \sum_{j'=1}^{k} z(\vec{x}, j')} \cdot \frac{\frac{1}{M \cdot L} R(\vec{x})\delta(\tilde{S}) + z(\vec{x}, j^*)}{\sum_{u \in \tilde{S}} y(\vec{x}, u)} \le$$
$$\le \alpha\beta \cdot \frac{\sum_{u \in V} y(\vec{x}, u)}{R(\vec{x}) + \sum_{j'=1}^{k} z(\vec{x}, j')} \cdot \frac{\frac{1}{M} R(\vec{x})\delta(S^*) + z(\vec{x}, j^*)}{\sum_{u \in S^*} y(\vec{x}, u)}$$
$$\le \alpha\beta .$$

The last inequality is derived from the last step of the proof. This concludes the proof. ☐

LEMMA 5.4. *For every $0 < \varepsilon \le 1$ and $\gamma > 0$ such that $\gamma \le \varepsilon$ and $\gamma \le \varepsilon M/\delta(S)$ for all $1 \le j \le k$ and $S \in \mathcal{F}_j$:*

$$(5.12) \quad lmax\left(P \cdot (\vec{x} + \gamma \cdot \mathbf{1}_{S,j})\right) - lmax\left(P \cdot \vec{x}\right) \le \gamma \cdot (1 + \varepsilon) \cdot N \cdot \frac{\frac{1}{M \cdot L} R(\vec{x})\delta(S) + z(\vec{x}, j)}{R(\vec{x}) + \sum_{j'=1}^{k} z(\vec{x}, j')}$$

$$(5.13) \quad lmin\left(C \cdot (\vec{x} + \gamma \cdot \mathbf{1}_{S,j})\right) - lmin\left(C \cdot \vec{x}\right) \ge \gamma \cdot (1 - \varepsilon/2) \cdot N \cdot \frac{\sum_{u \in S} y(\vec{x}, u)}{\sum_{u \in V} y(\vec{x}, u)} .$$

Lemma 5.4 can be derived immediately from Lemma 1 in [You01].

LEMMA 5.5. *For every $0 < \varepsilon \le 1$, $\gamma > 0$ satisfying the conditions as in Lemma 5.4, and scaling parameter $N = \varepsilon^{-1} \cdot \ln(k + 1)$, the output $\vec{x}$ of the algorithm satisfies:*

$$(5.14) \quad \sum_{j=1}^{k} \sum_{S \in \mathcal{F}_j : u \in S} x_{S,j} \ge 1 \qquad \forall u \in V$$

$$(5.15) \quad \sum_{S \in \mathcal{F}_j} x_{S,j} \le (1 + O(\varepsilon)) \cdot \alpha\beta \qquad \forall j = 1, \ldots, k$$

$$(5.16) \quad \frac{1}{M \cdot L} \sum_{j=1}^{k} \sum_{S \in \mathcal{F}_j} \delta(S) \cdot x_{S,j} \le (1 + O(\varepsilon)) \cdot \alpha\beta$$

Proof. By the stopping condition of the big while loop of the algorithm, it is clear that the covering constraint holds as in (5.14). Let us now consider both the packing constraints (5.15) and (5.16). Initially, since $\vec{x} = 0$, the starting value of $lmax(P \cdot \vec{x})$ is $\ln(k + 1)$. Lemma 5.4 implies that in every iteration, if $S \in \mathcal{F}_j$ for some j was chosen to be added fractionally to $\vec{x}$, then:

$$\frac{\Delta lmax(P \cdot \vec{x})}{\Delta lmin(C \cdot \vec{x})} \leq \frac{1+\varepsilon}{1-\varepsilon/2} \cdot \frac{\sum_{u \in V} y(\vec{x},u)}{R(\vec{x}) + \sum_{j'=1}^{k} z(\vec{x},j')} \cdot \frac{\frac{1}{M \cdot L} R(\vec{x}) \delta(S_j) + z(\vec{x},j)}{\sum_{u \in S_j} y(\vec{x},u)} .$$

The algorithm's choice of the cut S added to $\vec{x}$ gives that:

$$\frac{\Delta lmax(P \cdot \vec{x})}{\Delta lmin(C \cdot \vec{x})} \leq \alpha\beta \cdot \frac{1+\varepsilon}{1-\varepsilon/2} .$$

Therefore, since as long as the algorithm does not terminate $lmin(C \cdot \vec{x}) < N$ (by inequality (5.11)), and inequality (5.10) provides that:

$$\begin{aligned} N \cdot \max \Bigg\{ & \frac{1}{M \cdot L} \sum_{j=1}^{k} \sum_{S \in \mathcal{F}_j} \delta(S) \cdot x_{S,j} \ , \\ & \sum_{S \in \mathcal{F}_1} x_{S,1} \ , \ \ldots \ , \ \sum_{S \in \mathcal{F}_k} x_{S,k} \Bigg\} \leq \\ & \leq lmax(P \cdot \vec{x}) \\ & \leq \ln(k+1) + \alpha\beta \cdot \frac{1+\varepsilon}{1-\varepsilon/2} \cdot N + \\ & \quad \max \left\{ 1, \max_{1 \leq j \leq k} \max_{S \in \mathcal{F}_j} \{\delta(S)/M\} \right\} \gamma \\ & \leq \ln(k+1) + \alpha\beta \cdot \frac{1+\varepsilon}{1-\varepsilon/2} \cdot N + \varepsilon . \end{aligned}$$

The last inequality is derived by the conditions on γ. Since $N = \varepsilon^{-1} \cdot \ln(k+1)$ and $\alpha, \beta > 1$, we can conclude that:

$$\begin{aligned} \max \Bigg\{ & \frac{1}{M \cdot L} \sum_{j=1}^{k} \sum_{S \in \mathcal{F}_j} \delta(S) \cdot x_{S,j} \ , \\ & \sum_{S \in \mathcal{F}_1} x_{S,1} \ , \ \ldots \ , \ \sum_{S \in \mathcal{F}_k} x_{S,k} \Bigg\} \leq (1+O(\varepsilon)) \cdot \alpha\beta . \end{aligned}$$

This concludes the proof as (5.15) and (5.16) hold. □

LEMMA 5.6. *There is a choice of γ that satisfies the conditions in Lemma 5.4 such that the algorithm that outputs $\vec{x}$ runs in polynomial time.*

Proof. First, Lemma 5.3 implies that the algorithm never gets stuck. Second, let us bound the total number of big while iterations the algorithm makes. Consider the following potential function: $\sum_{j=1}^{k} \sum_{S \in \mathcal{F}_j} x_{S,j}$. Its initial value is 0, and its final value is at most $(1+O(\varepsilon)) \cdot \alpha\beta \cdot k$ by Lemma 5.5. In each iteration its value increases by γ, hence, after $(1+O(\varepsilon)) \cdot \alpha\beta \cdot k/\gamma$ big while iterations the algorithm stops.

Next, we prove that the running time of each iteration is polynomially bounded. We achieve this by bounding r, the number of internal while iterations (in step (6) of the algorithm), for every single big while iteration. From the proof of Lemma 5.3, we need to wait until the first time Y drops below $\sum_{u \in S^*} y(\vec{x},u)$. Recall that at any moment, for any vertex $u \in V$, $y(\vec{x},u) = e^{-N \cdot \gamma \cdot r_u}$, where r_u is the number of big while iterations in which a cut S that contains u was chosen. Thus,

$$\begin{aligned} y(\vec{x},u) &= e^{-N \cdot \gamma \cdot r_u} \\ &= e^{-\ln(k+1) \cdot \gamma/\varepsilon \cdot r_u} \\ &\geq \left(\frac{1}{\ln(k+1)} \right)^{r_u} \\ &\geq \left(\frac{1}{\ln(k+1)} \right)^{(1+O(\varepsilon)) \cdot \alpha\beta \cdot k/\gamma} . \end{aligned}$$

We used that $N = \varepsilon^{-1} \cdot \ln(k+1)$, $\gamma \leq \varepsilon$ and that r_u can be bounded by the total number of big while iterations. Therefore,

$$r \leq \frac{\ln(n) + (1+O(\varepsilon)) \cdot \alpha\beta \cdot k/\gamma \cdot \ln\ln(k+1)}{\ln(\beta)} ,$$

as initially $Y \leq n$. All that is left to prove is that one can choose $\gamma \geq 1/poly(n)$. The only restriction is that $\gamma \leq \varepsilon \cdot \min\{1, M/\delta(S)\}$ for every $S \in \mathcal{F}_j$ and for every $1 \leq j \leq k$. If the graph is unweighted it is obviously true that one can choose $\gamma \geq 1/poly(n)$, and standard rescaling techniques of the edge weights work. For example, one can throw away all edges of weight M/n^3 and rescale the remaining edge weights to be between 1 and n^3. □

Proof. [of Theorem 3.1] Follows immediately from Lemmas 5.5 and 5.6. □

6 Acknowledgements

The authors thank Yuval Peres and Konstantin Makarychev for helpful pointers to Freedman's inequality, and an anonymous reviewer for suggesting to us an alternative proof of Theorem 3.1, which we present in Section 5.2.

References

[AHKM11] M. Andrews, M. T. Hajiaghayi, H. Karloff, and A. Moitra. Capacitated metric labeling. In *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '11, pages 976–995. SIAM, 2011.

[ALN08] S. Arora, J. R. Lee, and A. Naor. Euclidean distortion and the sparsest cut. *J. Amer. Math. Soc.*, 21(1):1–21, 2008.

[AR98] Y. Aumann and Y. Rabani. An $O(\log k)$ approximate min-cut max-flow theorem and approximation algorithm. *SIAM J. Comput.*, 27(1), 1998.

[AR06] K. Andreev and H. Räcke. Balanced graph partitioning. *Theor. Comp. Sys.*, 39(6):929–939, 2006.

[ARV08] S. Arora, S. Rao, and U. V. Vazirani. Geometry, flows, and graph-partitioning algorithms. *Commun. ACM*, 51(10):96–105, 2008.

[Bar82] R. Barnes. An algorithm for partitioning the nodes of a graph. *SIAM. J. on Algebraic and Discrete Methods*, 3(4):541–550, 1982.

[BFK$^+$11] N. Bansal, U. Feige, R. Krauthgamer, K. Makarychev, V. Nagarajan, J. S. Naor, and R. Schwartz. Min-max graph partitioning and small set expansion. In *52nd Annual Symposium on Foundations of Computer Science*, pages 17–26. IEEE Computer Society, 2011.

[BMC$^+$12] P. Bodík, I. Menache, M. Chowdhury, P. Mani, D. A. Maltz, and I. Stoica. Surviving failures in bandwidth-constrained datacenters. In *Proceedings of the ACM SIGCOMM 2012 conference on Applications, technologies, architectures, and protocols for computer communication*, SIGCOMM '12, pages 431–442, New York, NY, USA, 2012. ACM.

[BVW88] E. R. Barnes, A. Vannelli, and J. Q. Walker. A new heuristic for partitioning the nodes of a graph. *SIAM. J. Discrete Mathematics*, 1(3):299–305, 1988.

[CGR08] S. Chawla, A. Gupta, and H. Räcke. Embeddings of negative-type metrics and an improved approximation to generalized sparsest cut. *ACM Transactions on Algorithms*, 4(2), 2008.

[CKN09] J. Cheeger, B. Kleiner, and A. Naor. A $(\log n)^{\Omega(1)}$ integrality gap for the sparsest cut SDP. In *FOCS*, pages 555–564. IEEE Computer Society, 2009.

[CS11] D. Chakrabarty and C. Swamy. Facility location with client latencies: linear programming based techniques for minimum latency problems. In *15th international conference on Integer programming and combinatoral optimization*, IPCO'11, pages 92–103. Springer-Verlag, 2011. Full version as arXiv:1009.2452.

[ENRS99] G. Even, J. Naor, S. Rao, and B. Schieber. Fast approximate graph partitioning algorithms. *SIAM J. Comput.*, 28(6):2187–2214, 1999.

[ENRS00] G. Even, J. Naor, S. Rao, and B. Schieber. Divide-and-conquer approximation algorithms via spreading metrics. *J. ACM*, 47(4):585–616, 2000.

[FF12] A. E. Feldmann and L. Foschini. Balanced Partitions of Trees and Applications. In *29th International Symposium on Theoretical Aspects of Computer Science (STACS 2012)*, volume 14, pages 100–111, Dagstuhl, Germany, 2012. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[FK06] U. Feige and R. Krauthgamer. A polylogarithmic approximation of the minimum bisection. *SIAM Review*, 48(1):99–130, 2006.

[Fre75] D. A. Freedman. On tail probabilities for martingales. *Ann. Probability*, 3:100–118, 1975.

[GVY93] N. Garg, V. V. Vazirani, and M. Yannakakis. Approximate max-flow min-(multi)cut theorems and their applications. *SIAM Journal on Computing*, 25:698–707, 1993.

[HMV92] S. Hadley, B. Mark, and A. Vannelli. An efficient eigenvector approach for finding netlist partitions. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 11(7), 1992.

[KLL$^+$13] T. C. Kwok, L. C. Lau, Y. T. Lee, S. Oveis Gharan, and L. Trevisan. Improved cheeger's inequality: analysis of spectral partitioning algorithms through higher order spectral gap. In *45th annual ACM symposium on Symposium on theory of computing*, STOC '13, pages 11–20. ACM, 2013.

[KNS09] R. Krauthgamer, J. Naor, and R. Schwartz. Partitioning graphs into balanced components. In *SODA*, pages 942–949, 2009.

[LLR95] N. Linial, E. London, and Y. Rabinovich. The geometry of graphs and some of its algorithmic applications. *Combinatorica*, 15(2), 1995.

[LM13] A. Louis and K. Makarychev. Approximation algorithm for sparsest k-partitioning. *CoRR*, abs/1306.4384, 2013.

[LN06] J. R. Lee and A. Naor. L_p metrics on the Heisenberg group and the Goemans-Linial conjecture. In *FOCS*, pages 99–108. IEEE Computer Society, 2006.

[LOT12] J. R. Lee, S. Oveis Gharan, and L. Trevisan. Multi-way spectral partitioning and higher-order cheeger inequalities. In *44th symposium on Theory of Computing*, pages 1117–1130. ACM, 2012.

[LR99] T. Leighton and S. Rao. Multicommodity max-flow min-cut theorems and their use in designing approximation algorithms. *J. ACM*, 46(6):787–832, 1999.

[LRTV11] A. Louis, P. Raghavendra, P. Tetali, and S. Vempala. Algorithmic extensions of cheeger's inequality to higher eigenvalues and partitions. In *14th international workshop and 15th international conference on Approximation, randomization, and combinatorial optimization: algorithms and techniques*, APPROX'11/RANDOM'11, pages 315–326. Springer-Verlag, 2011.

[LRTV12] A. Louis, P. Raghavendra, P. Tetali, and S. Vempala. Many sparse cuts via higher eigenvalues. In *44th Symposium on Theory of Computing*, pages 1131–1140. ACM, 2012.

[MMV12] K. Makarychev, Y. Makarychev, and A. Vijayaraghavan. Approximation algorithms for semi-random partitioning problems. In *Proceedings of the 44th symposium on Theory of Computing*, STOC '12, pages 367–384. ACM, 2012.

[R08] H. Räcke. Optimal hierarchical decompositions for congestion minimization in networks. In *Proceedings of the 40th annual ACM symposium on Theory of computing*, STOC '08, pages 255–264, 2008.

[RS10] P. Raghavendra and D. Steurer. Graph expansion and the unique games conjecture. In *STOC*, pages 755–764, 2010.

[RST10] P. Raghavendra, D. Steurer, and P. Tetali. Approximations for the isoperimetric and spectral profile of graphs and related parameters. In *STOC*, pages 631–640, 2010.

[RST12] P. Raghavendra, D. Steurer, and M. Tulsiani. Reductions between expansion problems. In *IEEE Conference on Computational Complexity*, pages 64–73, 2012.

[RW95] F. Rendl and H. Wolkowicz. A projection technique for partitioning the nodes of a graph. *Annals of Operations Research*, 58(3):155–179, 1995.

[San89] L. A. Sanchis. Multiple-way network partitioning. *IEEE Trans. Comput.*, 38(1):62–81, January 1989.

[You01] N. Young. Sequential and parallel algorithms for mixed packing and covering. In *Proceedings of the 42nd IEEE symposium on Foundations of Computer Science*, FOCS '01, pages 538–, Washington, DC, USA, 2001. IEEE Computer Society.