Combinatorial Approximation Algorithms

for MAXCUT using Random Walks

Satyen Kale[1]   C. Seshadhri[2*]

[1]Yahoo! Research, 4301 Great America Parkway, Santa Clara, CA 95054

[2]Sandia National Labs, Livermore, CA 94551

skale@yahoo-inc.com   scomand@sandia.gov

Abstract: We give the first combinatorial approximation algorithm for MAXCUT that beats the trivial 0.5 factor by a constant. The main partitioning procedure is very intuitive, natural, and easily described. It essentially performs a number of random walks and aggregates the information to provide the partition. We can control the running time to get an approximation factor-running time tradeoff. We show that for any constant $b > 1.5$, there is an $\tilde{O}(n^b)$ algorithm that outputs a $(0.5+\delta)$-approximation for MAXCUT, where $\delta = \delta(b)$ is some positive constant.

One of the components of our algorithm is a weak local graph partitioning procedure that may be of independent interest. Given a starting vertex i and a conductance parameter ϕ, unless a random walk of length $\ell = O(\log n)$ starting from i mixes rapidly (in terms of ϕ and ℓ), we can find a cut of conductance at most ϕ close to the vertex. The work done per vertex found in the cut is sublinear in n.

Keywords: MAXCUT, random walks, combinatorial algorithms, approximation algorithms.

1 Introduction

The problem of finding the maximum cut of a graph is a classical combinatorial optimization problem. Given a graph $G = (V, E)$, with weights w_{ij} on edges $\{i, j\}$, the problem is to partition the vertex set V into two sets L and R to maximize the weight of *cut edges* (these have one endpoint in L and the other in R). The value of a cut is the total weight of cut edges divided by the total weight. The largest possible value of this is $MaxCut(G)$. The problem of computing $MaxCut(G)$ was one of Karp's original NP-complete problems [12].

Therefore, polynomial-time approximation algorithms for MAXCUT were sought out, that would provide a cut with value at least $\alpha MaxCut(G)$, for some fixed constant $\alpha > 0$. It is easy to show that a random cut gives a 0.5-approximation for the MAXCUT. This was the best known for decades, until the seminal paper on semi-definite programming (SDP) by Goemans and Williamson [9]. They gave a 0.878-approximation algorithm, which is optimal for polynomial time algorithms under the Unique Games Conjecture [13,14]. Arora and Kale [3], and later, Steurer [21], gave an efficient near-linear-time implementation of the SDP algorithm for MAXCUT[1].

In spite of the fact that efficient, possibly optimal, approximation algorithms are known, there is a lot of interest in understanding what techniques are required to improve the 0.5-approximation factor. By "improve", we mean a ratio of the form $0.5 + \delta$, for some constant $\delta > 0$. The powerful technique of Linear Programming (LP) relaxations fails to improve the 0.5 factor. Even the use of strong LP-hierarchies to tighten relaxations does not help [7,17]. Recently, Trevisan [24] showed for the first time that a technique weaker than SDP relaxations can beat the 0.5-factor. He showed that the eigenvector corresponding to the smallest eigenvalue of the adjacency matrix can be used to approximate the MAXCUT to factor of 0.531. Soto [19] gave an improved analysis of the same algorithm that provides a better approximation factor of 0.6142. The running time[2] of this algorithm is $\tilde{O}(n^2)$.

---

* Work done when the author was a postdoctoral researcher at IBM Almaden Research Center.

[1]This was initially only proved for graphs in which the ratio of maximum to average degree was bounded by a polylogarithmic factor, but a linear-time reduction due to Trevisan [24] converts any arbitrary graph to this case.

[2]In this paper, we use the $\tilde{O}$ notation to suppress dependence on polylogarithmic factors.

All the previous algorithms that obtain an approximation factor better than 0.5 are not "combinatorial", in the sense that they all involve numerical matrix computations such as eigenvector computations and matrix exponentiations. It was not known whether combinatorial algorithms can beat the 0.5 factor, and indeed, this has been explicitly posed as an open problem by Trevisan [24]. Combinatorial algorithms are appealing because they exploit deeper insight into the structure of the problem, and because they can usually be implemented easily and efficiently, typically without numerical round-off issues.

1.1 Our contributions

1. In this paper, we achieve this goal of a combinatorial approximation algorithm for MaxCut. We analyze a very natural, simple, and combinatorial heuristic for finding the MaxCut of a graph, and show that it actually manages to find a cut with an approximation factor strictly greater than 0.5. In fact, we really have a suite of algorithms:

   Theorem 1.1 *For any constant $b > 1.5$, there is a combinatorial algorithm that runs in $\widetilde{O}(n^b)$ time and provides an approximation factor that is a constant greater than 0.5.*

   The running time/approximation factor trade-off curve is shown in Figure 1. A few representative numbers: in $\widetilde{O}(n^{1.6})$, $\widetilde{O}(n^2)$, and $\widetilde{O}(n^3)$ times, we can get approximation factors of 0.5051, 0.5155, and 0.5727 respectively. As b becomes large, this converges to the ratio of Trevisan's algorithm.

2. Even though the core of our algorithm is completely combinatorial, relying only on simple random walks and integer operations, the analysis of the algorithm is based on spectral methods. We obtain a combinatorial version of Trevisan's algorithm by showing two key facts: (a) the "flipping signs" random walks we use corresponds to running the power method on the graph Laplacian, and (b) a random starting vertex yields a good starting vector for the power method with constant probability. These two facts replace numerical matrix computations with the combinatorial problem of estimating certain probabilities, which can be done effectively by sampling and concentration bounds. This also allows improved running times since we can selectively

find portions of the graph and classify them.

3. A direct application of the partitioning procedure yields an algorithm whose running time is $\widetilde{O}(n^{2+\mu})$. To design the sub-quadratic time algorithm, we have to ensure that the random walks in the algorithm mix rapidly. To do this, we design a sort of a local graph partitioning algorithm of independent interest based on simple random walks of logarithmic length. Given a starting vertex i, either it finds a low conductance cut or certifies that the random walk from i has somewhat mixed, in the sense that the ratio of the probability of hitting any vertex j to its probability in the stationary distribution is bounded. The work done per vertex output in the cut is sublinear in n. The precise statement is given in Theorem 4.1. Previous local partitioning algorithms [1,2,20] are more efficient than our procedure, but can only output a low conductance cut, if the actual conductance of some set containing i is $O(1/\log n)$. In this paper, we need to be able to find low conductance cuts in more general settings, even if there is no cut of conductance of $O(1/\log n)$, and hence the previous algorithms are unsuitable for our purposes.

1.2 Related work

Trevisan [23] also uses random walks to give approximation algorithms for MaxCut (as a special case of unique games), although the algorithm only deals with the case when MaxCut is $1 - O(1/\text{poly}(\log n))$. The property tester for bipartiteness in sparse graphs by Goldreich and Ron [10] is a sublinear time procedure that uses random walks to distinguish graphs where $\text{MaxCut} = 1$ from $MaxCut \leqslant 1 - \varepsilon$. The algorithm, however, does not actually give an approximation to MaxCut. There is a similarity in flavor to Dinur's proof of the PCP theorem [8], which uses random walks and majority votes for gap amplification of CSPs. Our algorithm might be seen as some kind of belief propagation, where messages about labels are passed around.

For the special case of cubic and maximum degree 3 graphs, there has been a study of combinatorial algorithms for MaxCut [4,5,11]. These are based on graph theoretic properites and very different from our algorithms. Combinatorial algorithms for CSP (constraint satisfaction problems) based on LP relaxations have been studied in [6].

2   Algorithm overview and intuition

Let us revisit the greedy algorithm. We currently have a partial cut, where some subset S of the vertices have been classified (placed in either side of the cut). We take a new vertex $i \notin S$ and look at the edges of i incident to S. In some sense, each such edge provides a "vote" telling i where to go. Suppose there is such an edge (i, j), such that $j \in R$. Since we want to cut edges, this edge tells i to be placed in L. We place i accordingly to a majority vote, and hence the 0.5 factor.

Can we take that idea further, and improve on the 0.5 factor? Suppose we fix a source vertex i and try to classify vertices with respect to the source. Instead of just looking at edges (or paths of length 1), let us look at longer paths. Suppose we choose a length ℓ from some nice distribution (say, a binomial distribution with a small expectation) and consider paths of length ℓ from i. If there are many more even length paths to j than odd length paths, we put j in L, otherwise in R. This gives a partition of vertices that we can reach, and suggests an algorithm based on random walks. We hope to estimate the odd versus even length probabilities through random walks from i. This is a very natural idea and elegantly extends the greedy approach. We show that this can be used to beat the 0.5 factor by a constant.

One of the main challenges is to show that we do not need too many walks to distinguish these various probabilities. We also need to choose our length carefully. If it is too long, then the odd and even path probabilities may become too close to each other. If it is too short, then it may not be enough to get sufficient information to beat the greedy approach.

Suppose the algorithm detects that the probability of going from vertices i to j by an odd length path is significantly higher than an even length path. That suggests that we can be fairly confident that i and j should be on different sides of the cut. This constitutes the core of our algorithm, THRESHOLD. This algorithm classifies some vertices as lying on "odd" or "even" sides of the cut based on which probability (odd or even length paths) is significantly higher than the other. Significance is decided by a threshold that is a parameter to the algorithm. We show a connection between this algorithm and Trevisan's, and then we adapt his (and Soto's) analysis to show that one can choose the threshold carefully so that amount of work done per classified vertex is bounded, and the number of uncut edges is small. The search for the

right threshold is done by the FIND-THRESHOLD algorithm.

Now, this procedure leaves some vertices unclassified, because no probability is significantly larger than the other. We can simply recurse on the unclassified vertices, as long as the the cut we obtain is better than the trivial 0.5 approximate cut. This constitutes the SIMPLE algorithm. The analysis of this algorithm shows that we can bound the work done per vertex is at most $\widetilde{O}(n^{1+\mu})$ for any constant $\mu > 0$, and thus the overall running time becomes $\widetilde{O}(n^{2+\mu})$. This almost matches the running time of Trevisan's algorithm, which runs in $\widetilde{O}(n^2)$ time.

To obtain a sub-quadratic running time, we need to do a more careful analysis of the random walks involved. If the random walks do not mix rapidly, or, in other words, tend to remain within a small portion of the graph, then we end up classifying only a small number of vertices, even if we run a large number of these random walks. This is why we get the $\widetilde{O}(n^{1+\mu})$ work per vertex ratio.

But in this case, we can exploit the connection between fast mixing and high conductance [15,16,18] to conclude that there must be a low conductance cut which accounts for the slow mixing rate. To make this algorithmic, we design a local graph partitioning algorithm based on the same random walks as earlier. This algorithm, CUTORBOUND, finds a cut of (low) constant conductance if the walks do not mix, and takes only around $\widetilde{O}(n^{0.5+\mu})$ time, for any constant $\mu > 0$, per vertex found in the cut. Now, we can remove this low conductance set, and run SIMPLEon the induced subgraph. In the remaining piece, we recurse. Finally, we combine the cuts found randomly. This may leave up to half of the edges in the low conductance cut uncut, but that is only a small constant fraction of the total number of edges overall. This constitutes the BALANCE algorithm. We show that we spend only $\widetilde{O}(n^{0.5+\mu})$ time for every classified vertex, which leads to a $\widetilde{O}(n^{1.5+\mu})$ overall running time.

All of these algorithms are combinatorial: they only need random selection of outgoing edges, simple arithmetic operations, and comparisons. Although the analysis is technically involved, the algorithms themselves are simple and easily implementable.

We would like to stress some aspects of our presentation. Trevisan and Soto start with an n-dimensional vector x that has high Rayleigh quotient with the nor-

malized graph Laplacian (i.e. an approximate top eigenvector). This, they show, can be used to generate a cut that approximates the MaxCut value. It is possible to view our algorithm as a combinatorial procedure that produces this vector x, allowing us to leverage previous analyses. However, we prefer to depart from this presentation. We combine our combinatorial algorithm with the thresholding procedures of Trevisan and give one final stand alone algorithm. This highlights the core of the algorithm, which is a heuristic that one would naturally use to generalize the greedy algorithm[3]. We feel that our presentation brings out the intuition and combinatorial aspects behind the algorithm. The analysis is somewhat more complicated, since we need to use details of Trevisan's and Soto's work. However, since the main focus of this work is the elegance and power of combinatorial algorithms, we feel that this presentation is more meaningful.

3 The threshold cut

We now describe our core random walk based procedure to partition vertices. Some notation first. The graph G will have n vertices. All our algorithms will be based on lazy random walks on G with self-loop probability $1/2$. We define these walks now. Fix a length $\ell = O(\log n)$. At each step in the random walk, if we are currently at vertex j, then in the next step we stay at j with probability $1/2$. With the remaining probability $(1/2)$, we choose a random incident edge $\{j, k\}$ with probability proportional to w_{jk} and move to k. Thus the edge $\{j, k\}$ is chosen with overall probability $w_{jk}/2d_j$, where $d_j = \sum_{\{j,k\} \in E} w_{jk}$ is the (weighted) degree of vertex j. Let Δ be an upper bound on the maximum degree. By a linear time reduction of Trevisan [22,24], it suffices to solve MaxCut on graphs[4] where $\Delta = \text{poly}(\log n)$. We set m to be sum of weighted degrees, so $m := \sum_j d_j$. We note that by Trevisan's reduction, $m = \widetilde{O}(n)$, and thus running times stated in terms of m translate directly to the same polynomial in n.

The random walk described above is equivalent to flipping an unbiased coin ℓ times, and running a simple (non-lazy) random walk for h steps, where h is the number of heads seen. At each step of this simple random walk, an outgoing edge is chosen with probability

proportional to its weight. We call h the *hop-length* of the random walk, and we call a walk odd or even based on the parity of h.

We will denote the two sides of the cut by L and R. The parameters ε and μ are fixed throughout this section, and should be considered as constants. We will choose the length of the walk to be

$$\ell(\varepsilon, \mu) := \frac{\mu(\ln(4m/\delta^2))}{2(\delta + \varepsilon)}.$$

The reason for this choice will be explained later. Here δ is an arbitrarily small constant which controls the error tolerance. The procedure THRESHOLD takes as input a *threshold t*, and puts *some* vertices in one of two sets, $\mathcal{E}$ and $\mathcal{O}$, that are assumed to be global variables (i.e. different calls to Threshold update the same sets). We call vertices $j \in \mathcal{E} \cup \mathcal{O}$ *classified*. Once classified, a vertex is never re-classified. We perform a series of random walks to decide this. The number of walks will be a function of this threshold $w(t)$. We will specify this function later.

THRESHOLD
Input: Graph $G = (V, E)$.
Parameters: Starting vertex i, threshold t.
1. Perform $w(t)$ walks of length ℓ from i.
2. For every vertex j that is not classified:
   (a) Let $n_e(j)$ (resp. $n_o(j)$) be the number of even (resp. odd) length walks ending at j. Define
   $$\bar{y}_i(j) := \frac{n_e(j) - n_o(j)}{d_j w(t)}.$$
   (b) If $\bar{y}_i(j) > t$, put j in set $\mathcal{E}$. If $\bar{y}_i(j) < -t$, put it in set $\mathcal{O}$.

We normalize the difference of the number of even and odd walks by d_j to account for differences in degrees. This accounts for the fact that the stationary probability of the random walk at j is proportional to d_j. For the same reason, when we say "vertex chosen at random" we will mean choosing a vertex i with probability proportional to d_i. We now need some definitions.

Definition 3.1 (Work-to-output ratio.) *Let $\mathcal{A}$ be an algorithm that, in time T, classifies k vertices (into the sets $\mathcal{E}$ or $\mathcal{O}$). Then the work-to-output ratio of $\mathcal{A}$ is defined to be $\frac{T}{k}$.*

Definition 3.2(Good, Cross, Inc, Cut.) *Given two sets of vertices A and B, let Good(A, B) be the total*

---

weight of edges that have one endpoint in A and the other in B. Let $Cross(A, B)$ be the total weight of edges with only one endpoint in $A \cup B$. Let $Inc(A, B)$ be the total weight of edges incident on $A \cup B$. We set $Cut(A, B) := Good(A, B) + Cross(A, B)/2$.

Suppose we either put all the vertices in $\mathcal{E}$ in L or R, and the vertices in $\mathcal{O}$ in R or L respectively, retaining whichever assignment cuts more edges. Then the number of edges cut is at least $Cut(\mathcal{E}, \mathcal{O})$.

Definition 3.3(γ, δ, $w(t)$, σ, $f(\sigma)$.)

1. *We use γ and δ to denote sufficiently small constants. These will be much smaller than any other parameter of constant value. These are essentially precision parameters for our algorithms and theorems. The $O(\cdot)$ notation hides inverse polynomial dependence in these parameters.*

2. *For every vertex j, let p_j^ℓ be the probability of reaching j starting from i with an ℓ-length lazy random walk. Let α be an upper bound on $\max_j \frac{p_j^\ell}{d_j}$.*

3. *Define $w(t) := \frac{\kappa \ln(n) \max\{\alpha, t\}}{t^2}$, for a large enough constant κ.*

4. *Define $\sigma := 1 - (1 - \varepsilon)^{1 + \frac{1}{\mu}} - o(1)$, where the $o(1)$ term can be made as small as we please by setting δ, γ to be sufficiently small constants.*

5. *Define the function $f(\sigma)$ (c.f. [19]) as follows: here $\sigma_0 = 0.22815\ldots$ is a fixed constant.*

$$f(\sigma) = \begin{cases} 0.5 & \text{if } 1/3 < \sigma \leqslant 1 \\ \dfrac{-1 + \sqrt{4\sigma^2 - 8\sigma + 5}}{2(1 - \sigma)} & \text{if } \sigma_0 < \sigma \leqslant 1/3 \\ \dfrac{1}{1 + 2\sqrt{\sigma(1 - \sigma)}} & \text{if } 0 \leqslant \sigma \leqslant \sigma_0. \end{cases}$$

The constant σ_0 is simply the value of σ at which the last two expressions coincide, i.e. $\frac{-1+\sqrt{4\sigma_0^2-8\sigma_0+5}}{2(1-\sigma_0)} = \frac{1}{1+2\sqrt{\sigma_0(1-\sigma_0)}}$.

The parameter α measures how far the walk is from mixing, because the stationary probability of j is proportional to d_j. The function $f(\sigma) > 0.5$ when $\sigma < 1/3$, and this leads to an approximation factor greater than 0.5. Now we state our main performance bound for THRESHOLD.

Lemma 3.4 *Suppose $MaxCut \geqslant 1 - \varepsilon$. Then, there is a threshold t such that with constant prob-*

ability over the choice of a starting vertex i chosen at random, the following holds. The procedure THRESHOLD(i, t) outputs sets $\mathcal{E}$ and $\mathcal{O}$ such that $Cut(\mathcal{E}, \mathcal{O}) \geqslant f(\sigma)Inc(\mathcal{E}, \mathcal{O})$. Furthermore, the work-to-output ratio is bounded by $\widetilde{O}(\alpha \Delta m^{1+\mu} + 1/\alpha)$.

The main procedure of this section, FIND-THRESHOLD, is just an algorithmic version of the existential result of Lemma 3.4.

FIND-THRESHOLD
Input: Graph $G = (V, E)$.
Parameters: Starting vertex i, constant μ
  1. Initialize sets $\mathcal{E}$ and $\mathcal{O}$ to empty sets.
  2. For $t_r = (1 - \gamma)^r$, for $r = 0, 1, 2, \ldots$, as long as $t_r \geqslant \gamma/m^{1+\mu/2}$.
    (a) Run THRESHOLD(i, t_r).
    (b) If $Cut(\mathcal{E}, \mathcal{O}) \geqslant f(\sigma)Inc(\mathcal{E}, \mathcal{O})$ and $|\mathcal{E} \cup \mathcal{O}| \geqslant (\Delta t_r^2 n^{1+\mu} \log n)^{-1}$, output $\mathcal{E}$ and $\mathcal{O}$. Otherwise go to the next threshold.
  3. Output FAIL.

We are now ready to state the performance bounds for FIND-THRESHOLD.

Lemma 3.5 *Suppose $MaxCut \geqslant 1 - \varepsilon$. Let i be chosen at random. With constant probability over the choice of i and the randomness of FIND-THRESHOLD(i), the procedure FIND-THRESHOLD(i) succeeds and has a work to output ratio of $\widetilde{O}(\alpha \Delta m^{1+\mu} + 1/\alpha)$. Furthermore, regardless of the value of $MaxCut$ or the choice of i, the worst-case running time of FIND-THRESHOLD(i) is $\widetilde{O}(\alpha \Delta m^{2+\mu})$.*

The proofs of Lemmas 3.4 and 3.5 use results from Trevisan's and Soto's analyses [19,24]. The vectors we consider will always be n-dimensional, and should be thought of as an assignment of values to each of the n vertices in G. Previous analyses rest on the fact that a vector that has a large Rayleigh quotient (with respect to the graph Laplacian[5]) can be used to find good cuts. Call such a vector "good". These analyses show that partitioning vertices by *thresholding* over a good vector x yields a good cut. This means that for some threshold t, vertices j with $x(j) > t$ are placed in L and those with $x(j) < -t$ are placed in R. We would like to show that THRESHOLD is essentially performing such a thresholding on some good vector. We will construct a vector, somewhat like a distribution, related to THRESHOLD, and show that it is good. This

---
[5]For a vector x and matrix M, the Rayleigh quotient is $\frac{x^\top M x}{x^\top x}$.

requires an involved spectral analysis and is formalized in Lemma 3.7. With this in place, we use concentration inequalities and an adaptation of the techniques in [19] to connect thresholding to the cuts looked at by FIND-THRESHOLD.

In the following presentation, we first state Lemma 3.7. Then we will show how to prove Lemmas 3.4 and 3.5 using Lemma 3.7. This is rather involved, but intuitively should be fairly clear. It mainly requires understanding of the random process that THRESHOLD uses to classify vertices.

We need some definitions. Let A be the (weighted) adjacency matrix of G and d_i be the degree of vertex i. The (normalized) Laplacian of the graph is $\mathcal{L} = I - D^{-1/2}AD^{-1/2}$. Here D is the matrix where $D_{ii} = d_i$ and $D_{ij} = 0$ (for $i \neq j$). For a vector x and coordinate/vertex j, we use $x(j)$ to denote the jth coordinate of x (we do *not* use subscripts for coordinates of vectors). In [24] and [19], it was shown that vectors that have high Rayleigh quotients with $\mathcal{L}$ can be used to get a partition that cuts significant number of edges. Given a vector y, let us do a simple rounding to get partition vertices. We define the sets $P(y,t) = \{j \mid y(j) \geqslant t\}$ and $N(y,t) = \{j \mid y(j) \leqslant -t\}$. We refer to rounding of this form as *tripartitions*, since we divide the vertices into three sets. The following lemma, which is Lemma 4.2 from [19], an improvement of the analysis in [24], shows that this tripartition cuts many edges for some threshold:

Lemma 3.6 *([19]) Suppose $x^\top \mathcal{L}x \geqslant 2(1-\sigma)\|x\|^2$. Let $y = D^{-1/2}x$. Then, for some t (called* good*),*

$$\texttt{Cut}(P(y,t), N(y,t)) \;\geqslant\; f(\sigma)\texttt{Inc}(P(y,t), N(y,t)).$$

The algorithm of Trevisan is the following: compute the top eigenvector x of $\mathcal{L}$ (approximately), compute $y = D^{-1/2}x$, and find a good threshold t and the corresponding sets $P(y,t), N(y,t)$. Assign $P(y,t)$ or $N(y,t)$ to L and R (or vice-versa, depending on which assignment cuts more edges), and recurse on the remaining unclassified vertices.

The algorithms of this paper essentially mimic this process, except that instead of computing the top eigenvector, we use random walks. We establish a connection between random walks and the power method to compute the top eigenvector. Let $p_{i,j}^h$ be the probability that a length ℓ (remember that this is fixed) lazy random walk from i reaches j with hop-length h. Then define the vector q_i as follows: the jth coordi-

nate of q_i is

$$q_i(j) \;:=\; \frac{1}{\sqrt{d_j}} \left(\sum_{h \text{ even}} p_{i,j}^h - \sum_{h \text{ odd}} p_{i,j}^h \right)$$

$$= \; \frac{1}{\sqrt{d_j}} \sum_{h=0}^{\ell} (-1)^h p_{i,j}^h.$$

Note that THRESHOLD is essentially computing an estimate $\bar{y}_i(j)$ of $q_i(j)/\sqrt{d_j}$. For convenience, we will denote $D^{-1/2}q_i$ by y_i. This is the main lemma of this section.

Lemma 3.7 *Let $\delta > 0$ be a sufficiently small constant, and $\mu > 0$ be a (constant) parameter. If $\ell = \mu(\ln(4m/\delta^2))/[2(\delta + \varepsilon')]$, where $\varepsilon' = -\ln(1 - \varepsilon)$, then with constant probability over the choice of i, $\|q_i\|^2 = \Omega(1/m^{1+\mu})$, and*

$$q_i^\top \mathcal{L}q_i \;\geqslant\; 2e^{-(2+\frac{1}{\mu})\delta}(1 - \varepsilon)^{1+\frac{1}{\mu}}\|q_i\|^2, \qquad (1)$$

Although not at all straightforward, Lemma 3.7 and Lemma 3.6 essentially prove Lemma 3.4. To ease the flow of the paper, we defer these arguments to Section 3.1.

Lemma 3.7 is proved in two parts. In the first, we establish a connection between the random walks we perform and running the power method on the Laplacian:

Claim 3.8 *Let e_i be i^{th} standard basis vector. Then, we have $q_i = \frac{1}{2^\ell}\mathcal{L}^\ell\left(\frac{1}{\sqrt{d_i}}e_i\right)$.*

PROOF: *Note that $\mathcal{L}^\ell = (I - D^{-1/2}AD^{-1/2})^\ell = (D^{-1/2}(I - AD^{-1})D^{1/2})^\ell = D^{-1/2}(I - AD^{-1})^\ell D^{1/2}$. Hence,*

$$\mathcal{L}^\ell\left(\frac{1}{\sqrt{d_i}}e_i\right)$$

$$= D^{-1/2}(I - AD^{-1})^\ell e_i$$

$$= 2^\ell D^{-1/2} \sum_{h=0}^{\ell} (-1)^h \binom{\ell}{h} \left(\frac{1}{2}\right)^{\ell-h} \left(\frac{1}{2}AD^{-1}\right)^h e_i$$

$$= 2^\ell q_i$$

The last equality follows because the vector $\binom{\ell}{h}\left(\frac{1}{2}\right)^{\ell-h}\left(\frac{1}{2}AD^{-1}\right)^h e_i$ is the vector of probabilities of reaching different vertices starting from i in a walk of length ℓ with hop-length exactly h. We also used the facts that $D^{1/2}\frac{1}{\sqrt{d_i}}e_i = e_i$ and $D^{-1/2}e_j = \frac{1}{\sqrt{d_j}}e_j$. $\qquad \square$

In the second part, we show that with constant probability, a randomly chosen starting vertex yields a good starting vector for the power method, i.e., the vector q_i satisfies (1). This will require a spectral analysis. We need some notation first. Let the eigenvalues of $\mathcal{L}$ be $2 \geqslant \lambda_1 \geqslant \lambda_2 \geqslant \cdots \lambda_n = 0$, and let the corresponding (unit) eigenvectors be $v_1, v_2, \ldots v_n = D^{1/2}\vec{1}_V$. For a subset S of vertices, define $\mathrm{Vol}(S) = \sum_{i \in S} d_i$. Let $H = \{k : \lambda_k \geqslant 2e^{-\delta}(1-\varepsilon)\}$. Any vector x can be expressed in terms of the eigenvectors of $\mathcal{L}$ as $x = \sum_k \alpha_k v_k$. Define the norm $\|x\|_H = \sqrt{\sum_{k \in H} \alpha_k^2}$.

Let $(S, \bar{S})$ be the max-cut, where we use the convention $\bar{S} = V \setminus S$. Let $\mathrm{Vol}(S) \leqslant \mathrm{Vol}(V)/2$ and define $s := \mathrm{Vol}(S)$. Note that $m = \mathrm{Vol}(V)$. Since the max-cut has size at least $(1-\varepsilon)m/2$, we must have $s \geqslant (1-\varepsilon)m/2$. We set the vector $x = D^{1/2}y$ where $y = \frac{1}{s}\vec{1}_S - \frac{1}{m}\vec{1}_V$ where $\vec{1}_S$ is the indicator vector for S. We will need some preliminary claims before we can show (1).

Claim 3.9 $\|x\|_H^2 \geqslant \delta/m$.

PROOF. We have

$$
\begin{aligned}
x^\top \mathcal{L} x &= \sum_{i,j \in E} (y(i) - y(j))^2 \\
&= E(S, \bar{S}) \cdot \frac{1}{s^2} \\
&\geqslant \frac{(1-\varepsilon) \cdot (m/2)}{s^2} \\
&\geqslant \frac{(1-\varepsilon)m}{2s^2}
\end{aligned}
$$

Now[6], $\|x\|^2 = \frac{1}{s} - \frac{1}{m}$. Let $x = \sum_k \alpha_k v_k$ be the representation of x in the basis given by the v_k's, and let $a := \|x\|_H^2$. Then we have $\|x\|^2 = \sum_k \alpha_k^2$, and

$$
\begin{aligned}
x^\top \mathcal{L} x &= \sum_k \lambda_k \alpha_k^2 \\
&\leqslant 2 \sum_{k \in H} \alpha_k^2 + 2e^{-\delta}(1-\varepsilon) \sum_{k \notin H} \alpha_k^2 \\
&= 2a + 2e^{-\delta}(1-\varepsilon)\left(\frac{1}{s} - \frac{1}{m} - a\right).
\end{aligned}
$$

Combining the two bounds, and solving for a, we get the required bound for small enough δ. □

Claim 3.10 *With constant probability over the choice of i, we have $\|e_i\|_H^2 > \delta d_i/4m$.*

---

[6]This is easily seen using Pythagoras' theorem: since we have $D^{1/2}(\frac{1}{s}\vec{1}_S - \frac{1}{m}\vec{1}_V) \cdot D^{1/2}\vec{1}_V = 0$. This only uses the fact that $S \subseteq V$.

PROOF: Let $T := \{i \in S : \|\frac{1}{\sqrt{d_i}}e_i\|_H^2 < \frac{\delta}{4m}\}$, and let $t = \mathrm{Vol}(T)$. Our aim is to show that t is at most a constant fraction of s. For the sake of contradiction, assume $t \geqslant (1-\theta)s$, where $\theta = \delta(1-\varepsilon)/16$. Let

$$
z = D^{1/2}\left(\frac{1}{t}\vec{1}_T - \frac{1}{m}\vec{1}\right).
$$

We have

$$
\|x - z\|_H^2 \leqslant \|x - z\|^2 = \frac{1}{t} - \frac{1}{s} \leqslant \frac{2\theta}{s} \leqslant \frac{4\theta}{(1-\varepsilon)m} = \frac{\delta}{4m}.
$$

The second equality in the above uses the fact that $t \geqslant (1-\theta)s$ and $\theta < 1/2$. The third inequality follows from the fact that $s \geqslant (1-\varepsilon)m/2$. By the triangle inequality and Claim 3.9, we have

$$
\|z\|_H \geqslant \|x\|_H - \|x - z\|_H \geqslant \sqrt{\frac{\delta}{m}} - \sqrt{\frac{\delta}{4m}} = \sqrt{\frac{\delta}{4m}}.
$$

Now, we have $z = \sum_{i \in T}(\frac{d_i}{t})D^{1/2}(\frac{1}{d_i}e_i - \frac{1}{m}\vec{1}_V)$, so by Jensen's inequality, we get

$$
\begin{aligned}
\frac{\delta}{4m} &\leqslant \|z\|_H^2 \\
&\leqslant \sum_{i \in T} \frac{d_i}{t} \cdot \left\|D^{1/2}\left(\frac{1}{d_i}e_i - \frac{1}{m}\vec{1}_V\right)\right\|_H^2 \\
&= \sum_{i \in T} \frac{d_i}{t} \cdot \left\|D^{1/2}\left(\frac{1}{d_i}e_i\right)\right\|_H^2 \\
&< \frac{\delta}{4m},
\end{aligned}
$$

a contradiction. The equality in the above holds because $D^{1/2}\vec{1}_V$ has no component along the eigenvectors corresponding to H (this is an eigenvector itself, with eigenvalue 0).

Thus the set $S \setminus T$ has volume at least

$$
\theta s \geqslant \delta(1-\varepsilon)^2 m/32.
$$

Note that the sampling process, which chooses the initial vertex of the random walk by choosing a random edge and choosing a random end-point i of it, hits some vertex in $S \setminus T$ with probability at least $\frac{\mathrm{Vol}(S \setminus T)}{\mathrm{Vol}(V)} \geqslant \delta(1-\varepsilon)^2/32$, i.e. constant probability. □

At this point, standard calculations for the power method imply Lemma 3.7.

PROOF:(Of Lemma 3.7) From Claim 3.10, with constant probability $\|e_i\|_H^2 \geqslant \delta d_i/4m$. Let us assume this is case.

For convenience, define $\beta = \frac{(\delta + \varepsilon')}{\mu}$, so that the number of walks is $\ell = \frac{\ln(4m/\delta^2)}{2\beta}$. Now let

$$H' = \{i : \lambda_i \geqslant 2e^{-(\delta+\beta)}(1-\varepsilon)\}.$$

Write $\frac{1}{\sqrt{d_i}} e_i$ in terms of the v_k's as $\frac{1}{\sqrt{d_i}} e_i = \sum_k \alpha_k v_k$. Let $y_i = \mathcal{L}^\ell \frac{1}{\sqrt{d_i}} e_i = \sum_k \alpha_k \lambda_k^\ell v_k$. Note that $q_i = \frac{1}{2^\ell} y_i$. Then we have

$$\hat{y}_i^\top \mathcal{L} y_i = \sum_k \alpha_k^2 \lambda_k^{2\ell+1} \geqslant \sum_{k \in H'} \alpha_k^2 \lambda_k^{2\ell} \cdot 2e^{-(\delta+\beta)}(1-\varepsilon)$$

and

$$\|\hat{y}_i\|^2 = \sum_k \alpha_k^2 \lambda_k^{2\ell} = \sum_{k \in H'} \alpha_k^2 \lambda_k^{2\ell} \left[1 + \frac{\sum_{k \notin H'} \alpha_k^2 \lambda_k^{2\ell}}{\sum_{k \in H'} \alpha_k^2 \lambda_k^{2\ell}} \right].$$

We have

$$\frac{\sum_{k \notin H'} \alpha_k^2 \lambda_k^{2\ell}}{\sum_{k \in H'} \alpha_k^2 \lambda_k^{2\ell}} \leqslant \frac{\sum_{k \notin H'} \alpha_k^2 \lambda_k^{2\ell}}{\sum_{k \in H} \alpha_k^2 \lambda_k^{2\ell}}$$

$$\leqslant \frac{(2e^{-(\delta+\beta)}(1-\varepsilon))^{2\ell}}{\frac{\delta}{4m}(2e^{-\delta}(1-\varepsilon))^{2\ell}}$$

$$= \frac{4me^{-2\beta\ell}}{\delta}.$$

Thus,

$$\frac{\hat{y}_i^\top \mathcal{L} \hat{y}_i}{\|\hat{y}_i\|^2} \geqslant \frac{2e^{-(\delta+\beta)}(1-\varepsilon)}{1 + \frac{4me^{-2\beta\ell}}{\delta}}.$$

Observe that $\hat{y}_i$ is just a scaled version of q_i, so we can replace $\hat{y}_i$ by q_i above. For the denominator in the right, we would like to set that to be e^δ. Choosing $\ell = \frac{\ln(4m/\delta^2)}{2\beta}$, we get

$$q_i^\top \mathcal{L} q_i \geqslant 2e^{-(2\delta+\beta)}(1-\varepsilon)\|q_i\|^2$$

$$= 2e^{-(2+\frac{1}{\mu})\delta}(1-\varepsilon)^{1+\frac{1}{\mu}}\|q_i\|^2.$$

Since $\|e_i\|_H^2 \geqslant \delta d_i/4m$, we have

$$\sum_{k \in H} \alpha_k^2 = \left\| \frac{1}{\sqrt{d_i}} e_i \right\|_H^2 \geqslant \frac{\delta}{4m}.$$

This implies

$$\|q_i\|^2 = \frac{1}{2^{2\ell}} \|y_i\|^2 = \frac{1}{2^{2\ell}} \sum_k \alpha_k^2 \lambda_k^{2\ell} \geqslant \frac{1}{2^{2\ell}} \sum_{k \in H} \alpha_k^2 \lambda_k^{2\ell}.$$

By definition, for all $k \in H$, $\lambda_k \geqslant 2e^{-\delta}(1-\varepsilon)$. This gives a lower bound on the rate of decay of these coefficients, as the walk progresses.

$$\|q_i\|^2 \geqslant \frac{1}{2^{2\ell}} \sum_{k \in H} \alpha_k^2 (2e^{-\delta}(1-\varepsilon))^{2\ell}$$

$$\geqslant \frac{\delta}{4m} e^{-2\delta\ell}(1-\varepsilon)^{2\ell}$$

$$= \Omega\left(\frac{1}{4m^{1+\mu}} \right),$$

by our choice of $\ell = \frac{\ln(4m/\delta^2)}{2\beta}$. $\qquad \square$

3.1 Proofs of Lemmas 3.4 and 3.5

Both Lemma 3.5 and Lemma 3.4 follow directly from the following statement.

Lemma 3.11 *Let $w(t) = (c' \ln^2 /\gamma^2)(\alpha/t^2)$, where c' is a sufficiently large constant. Let C_r denote the set of vertices classified by* THRESHOLD(i, t_r). *The following hold with constant probability over the choice of i and the randomness of* THRESHOLD. *There exists a threshold $t_r = (1-\gamma)^r$ such that*

$$\sum_{j \in C_r} d_j = \Omega((t_r^2 m^{1+\mu} \log n)^{-1}).$$

Also, the tripartition generated satisfies Step 2(b) of FIND-THRESHOLD.

In this section, we will prove this lemma. But first, we show how this implies Lemmas 3.4 and 3.5.

PROOF (of Lemma 3.4) We take the threshold t_r given by Lemma 3.11. Since it satisfies Step 2(b) of FIND-THRESHOLD, $\mathsf{Cut}(\mathcal{E}, \mathcal{O}) \geqslant f(\sigma)\mathsf{Inc}(\mathcal{E}, \mathcal{O})$. To see the work to output ratio, observe that the work done is $\widetilde{O}(w(t_r)) = \widetilde{O}(\max(\alpha, t_r)/t_r^2)$. It is convenient to write this as $\widetilde{O}(\alpha/t_r^2 + 1/\alpha)$. The output is C_r. We have

$$\Delta|C_r| \geqslant \sum_{j \in C_r} d_j = \Omega\left(\frac{1}{t_r^2 m^{1+\mu} \log n} \right)$$

The output is at least 1. Therefore, the work per output is at most $\widetilde{O}(\alpha\Delta m^{1+\mu} + 1/\alpha)$. $\qquad \square$

PROOF (of Lemma 3.5) The running time when there is failure is easy to see. The running time upto round r is $\widetilde{O}(\sum_{j \leqslant r} \max(\alpha, t_j)/t_j^2) = \widetilde{O}(\alpha/t_r^2 + 1/\alpha)$. Since $r^* = 1/n^{1+\mu/2}$ and $\alpha \leqslant 1/n^2$, we get the desired bound. By Lemma 3.11, we know that FIND-THRESHOLD succeeds with high probability. We have some round r where FIND-THRESHOLD will terminate

(satisfying the conditions of Step 2(b)). The work to output ratio analysis is the same as the previous proof, and is at most $\widetilde{O}(\alpha \Delta m^{1+\mu} + 1/\alpha)$. $\qquad\square$

We will first need some auxilliary claims that will help us prove Lemma 3.11. The first step is the use concentration inequalities to bound the number of walks required to get coordinates of y_i. As mentioned before, we designate the coordinates of q_i by $q_i(j)$. The p_i vector is the probability vector of the random walk (without charges) for ℓ steps. In other words, using the notation $i\overset{\text{even}}{\longrightarrow}j$ (resp. $i\overset{\text{odd}}{\longrightarrow}j$) to denote the events that an even (resp. odd) length walk from i reaches j, we have

$$y_i(j) := (\mathbf{Pr}[i\overset{\text{even}}{\longrightarrow}j] - \mathbf{Pr}[i\overset{\text{odd}}{\longrightarrow}j])/d_j$$

and

$$p_i(j) := \mathbf{Pr}[i\overset{\text{even}}{\longrightarrow}j] + \mathbf{Pr}[i\overset{\text{odd}}{\longrightarrow}j].$$

This clearly shows that the random walks performed by THRESHOLD are being used to estimate coordinates of q_i. The following claim shows how many w walks are required to get good a approximation of coordinates q_i.

Claim 3.12 *Suppose w walks are performed. Let c be a sufficiently large constant and $1/\ln n < \gamma < 1$. The following hold with probability at least $> 1 - n^{-4}$.*

- *If $w \geqslant (c\ln n/\gamma^2)(\max(\alpha, t)/t^2)$, then we can get an estimate $\bar{y}_i(j)$ such that $\sqrt{d_i}|\bar{y}_i(j) - y_i(j)| \leqslant \gamma t$.*

- *If $w \geqslant (c\ln n/\gamma^2)m^{1+\mu}$, then we can get an estimate $\bar{y}_i(j)$ such that $\sqrt{d_j}|\bar{y}_i(j) - q_i(j)| \leqslant \beta_j$, where $\beta_j := \sqrt{\frac{\gamma^2 \max\{p_j, 1/m^{1+\mu}\}}{m^{1+\mu}}}$.*

PROOF We define a vector of random variables X_k, one for each walk. Define random variables $X_k(j)$ as follows:

$$X_k(j) = \begin{cases} 1 & \text{walk } k \text{ ends at } j \text{ with even hops} \\ -1 & \text{walk } k \text{ ends at } j \text{ with odd hops} \\ 0 & \text{walk } k \text{ doesn't end at } j \end{cases}$$

Note that $\mathbb{E}[X_k(j)] = y_i(j)d_j$, and $\mathbf{Var}[X_k(j)] = p_j$. Our estimate $\bar{y}_i(j)$ will be $\frac{1}{w}\sum_k X_k(j)$. Observing that $|X_k(j)| \leqslant 1$, Bernstein's inequality implies that for any $\beta > 0$,

$$\mathbf{Pr}\left[\left|\frac{1}{wd_j}\sum_{k=1}^{w} X_k(j) - y_i(j)\right| > \beta\right]$$
$$\leqslant 2\exp\left(-\frac{3w\beta^2 d_j^2}{6p_i(j) + 2\beta d_j}\right).$$

For the first part, we set $\beta = \gamma t/\sqrt{d_j}$. For a sufficiently large c, We get that the exponent is at least $4\ln n$, and hence the probability is at most $1/n^4$. For the second part, we set $\beta = \beta_j/\sqrt{d_j}$. Note that if $p_j < 1/m^{1+\mu}$, then $\beta_j < 1/m^{1+\mu}$. So, the exponent is at least $4\ln n$, completing the proof. $\qquad\square$

We need to find a vector with a large Rayleigh quotient that can be used in Lemma 3.6. We already have a candidate vector q_i. Although we get a very good approximation of this, note that the order of vertices in an approximation can be very far from q_i. Nonetheless, the following lemma allows us to do so.

Claim 3.13 *Let x be a vector such that $x^\top \mathcal{L}x \geqslant (2 - \varepsilon)\|x\|^2$. Then, if x' is a vector such that $\|x - x'\| < \delta\|x\|$, then*

$$\|x'\|^2 \geqslant (1 - 3\delta)\|x\|^2$$

and

$$x'^\top \mathcal{L}x' \geqslant (2 - \varepsilon - 12\delta)\|x'\|^2.$$

PROOF We have

$$x'^\top \mathcal{L}x' - x^\top \mathcal{L}x = x'^\top \mathcal{L}x' - x'^\top \mathcal{L}x + x'^\top \mathcal{L}x - x^\top \mathcal{L}x$$
$$= (x' - x)^\top \mathcal{L}(x + x').$$

Thus,

$$|x'^\top \mathcal{L}x' - x^\top \mathcal{L}x| \leqslant (\|x'\| + \|x\|) \cdot \|\mathcal{L}\| \cdot \|x - x'\|$$
$$\leqslant (2 + \delta)\|x\| \cdot 2 \cdot \delta\|x\|$$
$$\leqslant 6\delta\|x\|^2.$$

Furthermore,

$$|\|x'\|^2 - \|x\|^2| \leqslant (\|x'\| + \|x\|) \cdot \|x - x'\|$$
$$\leqslant (2 + \delta)\|x\| \cdot \delta\|x\|$$
$$\leqslant 3\delta\|x\|^2.$$

Thus, we have

$$x'^\top \mathcal{L}x' \geqslant x^\top \mathcal{L}x - 6\delta\|x\|^2$$
$$\geqslant (2 - \varepsilon - 6\delta)\|x\|^2$$
$$\geqslant \frac{(2 - \varepsilon - 6\delta)}{(1 + 3\delta)}\|x'\|^2$$
$$\geqslant (2 - \varepsilon - 12\delta)\|x'\|^2.$$

$\qquad\square$

Now we prove Lemma 3.11.

PROOF: Our cutting procedure is somewhat different from the sweep cut used in [24]. The most naive

cut algorithm would take q_i and perform a sweep cut. Lemma 3.7 combined Lemma 3.6 would show that we can get a good cut. Unfortunately, we are using an approximate version of y_i ($\bar{y}_i$) for this purpose. Nonetheless, Claim 3.12 tells us that we can get good estimates of y_i, so $\bar{y}_i$ is close to y_i. Claim 3.13 tells us that $\bar{y}_i$ is good enough for all these arguments to go through (since Lemma 3.6 only requires a bound on the Rayleigh quotient).

Our algorithm FIND-THRESHOLD is performing a geometric search for the right threshold, invoking THRESHOLD many times. In each call of the THRESHOLD, let estimate vector $\bar{y}_i^{(r)}$ be generated. Using these, we will construct a vector $\widetilde{y}_i$. This construction is not done by the algorithm, and is only a thought experiment to help us analyze FIND-THRESHOLD.

Initially, all coordinates of $\widetilde{y}_i$ are not defined, and we incrementally set values. We will call THRESHOLD(i, t_r) in order, just as FIND-THRESHOLD. In the call to THRESHOLD(i, t_r), we observe that vertices which are classified. These are the vertices j for which $\bar{y}_i^{(r)}(j) > t_r$ and which have not been classified before. For all such j, we set $\widetilde{y}_i(j) := t_r$. We then proceed to the next call of THRESHOLD and keep continuing until the last call. After the last invocation of THRESHOLD, we simply set any unset $\widetilde{y}_i(j)$ to 0.

Claim 3.14 $\|D^{1/2}\widetilde{y}_i - q_i\| \leqslant 7\gamma\|q_i\|$

PROOF Suppose $y_i(j) > t_r(1 + 4\gamma)$. Note that $\|\bar{y}_i^{(r-1)}(j) - q_i(j)\| \leqslant \gamma t_{r-1}/\sqrt{d_j}$. Therefore,

$$
\begin{aligned}
\bar{y}_i^{(r-1)}(j) &> t_r(1 + 4\gamma) - \gamma t_{r-1} \\
&> t_r(1 + 4\gamma) - \gamma(1 + 2\gamma)t_r \\
&\geqslant t_r(1 + 2\gamma) \\
&\geqslant t_{r-1}.
\end{aligned}
$$

So $\widetilde{y}_i(j)$ must be set in round $r - 1$, if not before. If $\widetilde{y}_i(j)$ remains unset to the end (and is hence 0), then we have $y_i(j) \leqslant t_r(1 + 4\gamma)$. This bound implies that $q_i(j) \leqslant 2\gamma/m^{1+\mu/2}$. The total contribution of all these coordinates to the difference $\|D^{1/2}\widetilde{y}_i - q_i\|^2$ is at most $4\gamma^2/m^{1+\mu} \leqslant 4\gamma^2\|q_i\|^2$.

Suppose $\widetilde{y}_i(j)$ is set in round r to t_r. This means that $\bar{y}_i^{(r)}(j) > t_r$. By the choice of $w(t_r)$ and Claim 3.12, $\sqrt{d_j}|\bar{y}_i^{(r)}(j) - y_i(j)| \leqslant \gamma t_r$. Therefore,

$$
|\sqrt{d_j}\bar{y}_i^{(r)}(j) - q_i(j)| \leqslant \gamma t_r \leqslant 2\gamma q_i(j)
$$

$$
\begin{aligned}
\implies \quad \sqrt{d_j}\bar{y}_i^{(r)}(j) &\leqslant (1 + 2\gamma)q_i(j) \\
\implies \quad \sqrt{d_j}\widetilde{y}_i(j) &= \sqrt{d_j}t_r \leqslant (1 + 2\gamma)q_i(j).
\end{aligned}
$$

Combining with the first part, we get

$$
|\sqrt{d_j}\widetilde{y}_i(j) - q_i(j)| \leqslant 5\gamma q_i(j).
$$

$\square$

We now observe that sweep cuts in $\widetilde{y}_i$ generate *exactly* the same classifications that THRESHOLD(i, t_r) outputs. Therefore, it suffices to analyze sweep cuts of $\widetilde{y}_i$. We need to understand why there are thresholds that cut away many vertices. Observe that the coordinates of $\widetilde{y}_i$ are of the form $(1 - \gamma)^r$. This vector partitions all vertices in a natural way. For each r, define $R_r := \{j | \widetilde{y}_i(j) = t_r\}$. Call r *sparse*, if

$$
\left(\sum_{j \in R_r} d_j\right) t_r^2 \leqslant \frac{\gamma^3}{m^{1+\mu}\log n}.
$$

Otherwise, it is *dense*. Note that a dense threshold exactly satisfies the condition in Lemma 3.11. Abusing notation, we call a vertex j sparse if $j \in R_r$, such that r is sparse. Similarly, a threshold t_r is sparse if r is sparse. We construct a vector $\widehat{y}_i$. If $j \in R_r$, for r sparse, then $\widehat{y}_i(j) := 0$. Otherwise, $\widehat{y}_i(j) := \widetilde{y}_i(j)$.

Claim 3.15 $\|D^{1/2}(\widehat{y}_i - \widetilde{y}_i)\| \leqslant 2\gamma\|q_i\|$

PROOF

$$
\begin{aligned}
\|D^{1/2}(\widehat{y}_i - \widetilde{y}_i)\|^2 &= \sum_{j:\widehat{y}_i(j)=0} d_j\widetilde{y}_i(j)^2 \\
&= \sum_{r:r \text{ sparse}} \sum_{j \in R_r} d_j\widetilde{y}_i(j)^2 \\
&= \sum_{r:r \text{ sparse}} \sum_{j \in R_r} d_j t_r^2 \\
&\leqslant \frac{4\log n}{\gamma} \cdot \frac{\gamma^3}{m^{1+\mu}\log n} \\
&= \frac{4\gamma^2}{m^{1+\mu}\log n} \\
&\leqslant \gamma^2 4\|q_i\|^2.
\end{aligned}
$$

$\square$

Let us now deal with the vector $\widehat{y}_i$ and perform the sweep cut of [24]. All coordinates of $\widehat{y}_i$ are at most 1. We choose a threshold t at random: we select t^2 uniformly at random[7] from $[0, 1]$. We do a rounding

---

[7]Both [24] and [19] actually select t uniformly at random, and use $\sqrt{t}$ as a threshold. We do this modified version because it is more natural, for our algorithm, to think of the threshold as a lower bound on the probabilities we can detect.

to get the vector $z_t \in \{-1, 0, 1\}^n$:

$$z_t(j) = \begin{cases} 1 & \text{if } \widehat{y}_{ii}(j) \geqslant t \\ -1 & \text{if } \widehat{y}_{ii}(j) \leqslant -t \\ 0 & \text{if } |\widehat{y}_{ii}(j)| < t \end{cases}$$

The non-zero vertices in z_t are classified accordingly. A *cut* edge is one both of whose endpoints are non-zero and of opposite size. A *cross* edge is one where only one endpoint is zero. This classifying procedure is shown to cut a large fraction of edges. By Lemma 3.7, we have $q_i^\top \mathcal{L} q_i \geqslant 2(1 - \bar{\varepsilon}) \|q_i\|^2$ (where $\bar{\varepsilon}$ is some function of ε and μ). By Claims 3.14, 3.15 and Claim 3.13, $(D^{1/2}\widehat{y}_{ii})^\top \mathcal{L}(D^{1/2}\widehat{y}_{ii}) \geqslant 2(1 - \bar{\varepsilon} - c\gamma)\|D^{1/2}\widehat{y}_{ii}\|^2$. Then, by Lemma 3.6, there are good thresholds for y_i. It remains to prove the following claim.

Claim 3.16 *There exit dense and good thresholds for $\widetilde{y}_i$.*

PROOF We follow the analysis of [19]. We will perform sweep cuts for both $\widetilde{y}_i$ and y_i and follow their behavior. First, let take the sweep cut over y_i. Consider the indicator random variable $C(j, k)$ (resp. $X(j, k)$) that is 1 if edge (j, k) is a cut (resp. cross) edge. It is then show that $\mathbb{E}[C(j, k) + \beta X(j, k)] \geqslant \beta(1 - \beta)(\widehat{y}_i(j) - \widehat{y}_i(k))^2$, where the expectation is over the choice of the threshold t. Let us define a slight different choice of random thresholds. As before t^2 is chosen uniformly at random from $[0, 1]$. Then, we find the smallest t_r such that r is dense and $t_r \geqslant t$. We use this $t^* := t_r$ as the threshold for the cut. Observe that this gives the *same* distribution over cuts as the original and *only* selects dense thresholds. This is because in $\widehat{y}_i$ all non-dense vertices are set to 0. All thresholds strictly in between two consective dense t_r's output the same classification. The expectations of $C(j, k)$ and $X(j, k)$ are still the same.

We define analogous random variables $C'(j, k)$ and $X'(j, k)$ for $\widetilde{y}_i$. We still use the distribution over dense thresholds as described above. When both j and k are dense, we note that $C'(j, k) = C(j, k)$ and $X'(j, k) = X(j, k)$. This is because if t falls below, say, $\widetilde{y}_i(j)$ (which is equal to $\bar{y}_i(j)$), then j will be cut. Even though $t^* > t$, it will not cross $\widetilde{y}_i(j)$, since j is dense. So, we have $\mathbb{E}[C'(j, k) + \beta X'(j, k)] = \mathbb{E}[C(j, k) + \beta X(j, k)]$.

If both j and k are not dense, then $C'(j, k) = X'(j, k) = 0$. Therefore,

$$\mathbb{E}[C(j, k) + \beta X(j, k)] \geqslant \mathbb{E}[C'(j, k) + \beta X'(j, k)].$$

That leaves the main case, where k is dense but j is not. Note that $\mathbb{E}[C(j, k)] = 0$, since $y_i(j) = 0$. We

have

$$\mathbb{E}[X(j, k)] = \widehat{y}_{ii}(k)^2 = \widetilde{y}_i(k)^2.$$

If $|\widetilde{y}_i(j)| \leqslant |\widetilde{y}_i(k)|$, then

$$\mathbb{E}[X'(j, k)] = \widetilde{y}_i(k)^2 - \widetilde{y}_i(j)^2.$$

If $|\widetilde{y}_i(j)| \leqslant |\widetilde{y}_i(k)|$, then

$$\mathbb{E}[X'(j, k)] \geqslant 0 \geqslant \widetilde{y}_i(k)^2 - \widetilde{y}_i(j)^2.$$

So, we can bound

$$\mathbb{E}[X'(j, k)] \geqslant \mathbb{E}[X(j, k)] - \widetilde{y}_i(j)^2$$

and

$$\mathbb{E}[C'(j, k) + \beta X'(j, k)] \geqslant \beta(1 - \beta)(\widetilde{y}_i(j) - \widetilde{y}_i(k))^2 - \beta \widetilde{y}_i(j)^2.$$

Summing over all edges, and applying the bound in Lemma 4.2 of 3.6 for the non-prime random variables (dealing with $\widehat{y}_i$), we get

$$\mathbb{E}\left[\sum_{(j,k)} C'(j, k) + \beta X'(j, k)\right]$$

$$\geqslant \mathbb{E}\left[\sum_{(j,k)} C(j, k) + \beta X(j, k)\right] - \beta \sum_{j \text{ sparse}} d_j \widetilde{y}_i(j)^2$$

$$\geqslant \beta(1 - \beta) \sum_{(j,k) \text{ edge}} (\widehat{y}_i(j) - \widehat{y}_i(k))^2 - \beta\gamma^2 \|D^{1/2}\widetilde{y}_i\|^2$$

$$= \beta(1 - \beta)(D^{1/2}\widehat{y}_i)^\top \mathcal{L}(D^{1/2}\widehat{y}_i) - \beta\gamma^2 \|D^{1/2}\widetilde{y}_i\|^2$$

$$\geqslant 2(1 - \hat{\sigma})\beta(1 - \beta)\|D^{1/2}y_i\|^2 - 4\beta(1 - \beta)\gamma^2 \|D^{1/2}y_i\|^2$$

$$\geqslant 2(1 - \sigma)\beta(1 - \beta)\|D^{1/2}\widetilde{y}_i\|^2.$$

The second last step comes from the bound on $(D^{1/2}\widehat{y}_{ii})^\top \mathcal{L}(D^{1/2}\widehat{y}_{ii})$ we have found, and the observation that β will always be set to less than $1/2$. We have $1 - \hat{\sigma} = e^{-(2\delta + \mu)}(1 - \varepsilon) - O(\gamma)$ (based on Lemma 3.7. Since $|\sigma - \hat{\sigma}| = O(\gamma)$, we get σ as given in Lemma 3.5. Because of the equations above, the analysis of [19] shows that the randomly chosen threshold t^* has the property that

$$\texttt{Cut}(P(\widetilde{y}_i, t^*), N(\widetilde{y}_i, t^*))$$
$$\geqslant f(\sigma)\texttt{Inc}(P(\widetilde{y}_i, t^*), N(\widetilde{y}_i, t^*)).$$

Therefore, some threshold satisfies the condition 2(b) of FIND-THRESHOLD. Note that the thresholds are chosen over a distribution of dense thresholds. Hence, there is a good and dense threshold. □

□

4 CUTORBOUND **and local partitioning**

We describe our local partitioning procedure CU-
TORBOUND which is used to get the improved run-
ning time. We first set some notation. For a subset
of vertices $S \subseteq V$, define $\bar{S} = V \setminus S$, and let $E(S, \bar{S})$
be the *set* of edges crossing the cut $(S, \bar{S})$. Define
the *weight* of S to be $\omega(S) = 2\text{Vol}(S)$, to account
for the self-loops of weight $1/2$: we assume that each
vertex has a self-loop of weight d_i, and the random
walk simply chooses one edge with probability propor-
tional to its weight. For convenience, given a vertex
j, $\omega(j) = \omega(\{j\}) = 2d_j$. For a subset of edges $F \subseteq E$,
let $\omega(F) = \sum_{e \in F} w_e$. The conductance of the set S,
ϕ_S, is defined to be $\phi_S = \frac{\omega(E(S, \bar{S}))}{\min\{\omega(S), \omega(\bar{S})\}}$.

CUTORBOUND

Input: Graph G.

Parameters: Starting vertex i, constants τ, ζ,
length ℓ such that $\ell \geqslant \ln(m)/\zeta$.

Derived parameters:

  1. Define $\alpha := m^{-\tau}$, constant ϕ chosen to satisfy

$$-\log\left(\frac{1}{2}(\sqrt{1-2\phi} + \sqrt{1+2\phi})\right) = \zeta\tau,$$

    $w := \lceil 30\ell^2 \ln(n)/\alpha \rceil$, and $b := \left\lceil \frac{\ell}{2(1-2\phi)\alpha} \right\rceil$.

  2. Run w random walks of length ℓ from i.

  3. For each length $l = 0, 1, 2, \ldots, \ell$:

  (a) For any vertex j, let w_j be the number of walks
     of length l ending at j. Order the vertices in
     decreasing order of the ratio of w_j/d_j, breaking
     ties arbitrarily.

  (b) For all $k \leqslant b$, compute the conductance of the
     set of top k vertices in this order.

  (c) If the conductance of any such set is less than
     ϕ, stop and output the set.

  4. Declare that $\max_j \frac{p_j}{2d_j} \leqslant 256\alpha$.

The main theorem of this section is:

Theorem 4.1 *Suppose a lazy random walk is run from
a vertex i for $\ell \geqslant \ln(m)/\zeta$ steps, for some constant
ζ. Let p^ℓ be the probability distribution induced on
the final vertex. Let $\alpha = m^{-\tau}$, for constant $\tau < 1$,
be a given parameter so that $\zeta\tau < 1/8$, and let ϕ be
chosen to satisfy $-\log(\frac{1}{2}(\sqrt{1-2\phi} + \sqrt{1+2\phi})) = \zeta\tau$.
Then, there is an algorithm CUTORBOUND, that with
probability $1 - o(1)$, in $O(\Delta \log^4(n)/\alpha)$ time, finds a
cut of conductance less than ϕ, or declares correctly
that $\max_j \frac{p_j^\ell}{2d_j} \leqslant 256\alpha$.*

We provide a sketch before giving the detailed proof.
We use the Lovász-Simonovits curve technique [15].
For every length $l = 0, 1, \ldots, \ell$, let p^l be the proba-
bility vector induced on vertices after running a ran-
dom walk of length l. The Lovász-Simonovits curve
$I^l : [0, 2m] \rightarrow [0, 1]$ is constructed as follows. Let
$j_1, j_2, \ldots, j_n$ be an ordering of the vertices such that

$$\frac{p_{j_1}^l}{\omega(j_1)} \;\geqslant\; \frac{p_{j_2}^l}{\omega(j_2)} \;\geqslant\; \cdots \;\geqslant\; \frac{p_{j_1}^l}{\omega(j_n)}.$$

For $k \in \{1, \ldots, n\}$, define the set $S_k^l = \{j_1, j_2, \ldots, j_k\}$. For convenience, we define $S_0^l = \emptyset$, the
empty set. For a subset of vertices S, and a probability
vector p, define $p(S) = \sum_{i \in S} p_i$. Then, we define the
curve I^l at the following points: $I^l(\omega(S_k^l)) := p^l(S_k^l)$,
for $k = 0, 1, 2, \ldots, n$. Now we complete the curve I^l
by interpolating between these points using line seg-
ments. Note that this curve is concave because the
slopes of the line segments are decreasing. Also, it is
an increasing function. Lovász and Simonovits prove
that as l increases, I^l "flattens" out, at a rate gov-
erned by the conductance. A flatter I^l means that
the probabilities at vertices are more equal (slopes are
not very different), and hence the walk is mixing.

Roughly speaking, the procedure CUTORBOUND
only looks the portion of I^l upto S_b^l, since it only tries
to find sweep cuts among the top b vertices. We would
like to argue that if CUTORBOUND is unsuccessful in
finding a low conductance cut there, the maximum
probability should be small. In terms of the I^ls, this
means that the portion up to S_b^l flattens out rapidly.
In some sense, we want to prove versions of theorems
in [15] that only talk about a prefix of the I^l curves.

The issue now is that it is not possible to compute
the p_j^l's (and I^l) exactly since we only use random
walks. We run walks of length l and get an empirical
distribution $\tilde{p}^l$. We define $\tilde{I}^l$ to be the correspond-
ing Lovász-Simonovits curve corresponding to $\tilde{p}^l$. If
we run sufficiently many random walks and aggregate
them to compute $\tilde{p}_j^l$, then concentration bounds im-
ply that p_j^l is close to $\tilde{p}_j^l$ (when p_j^l is large enough).
Ideally, this should imply that the behavior of $\tilde{I}^l$ is
similar to I^l. There is a subtle difficulty here. The *or-
der* of vertices with respect to p^l and $\tilde{p}^l$ could be very
different, and hence prefixes in the I^l and $\tilde{I}^l$ could be
dealing with different subsets of vertices. Just because
I^l is flattening, it is not obvious that $\tilde{I}^l$ is doing the
same.

Nonetheless, because for large p_j^l's, $\tilde{p}_j^l$ is a good

approximation, some sort of flattening happens for $\widetilde{I}^l$. We give some precise expressions to quantify this statement. Suppose CUTORBOUND is unable to find a cut of conductance ϕ. Then we show that for any $x \in [0, 2m]$, if $\hat{x} = \min\{x, 2m - x\}$,

$$\widetilde{I}^l(x) \leqslant \frac{e^{3\eta}}{2}(\widetilde{I}^{l-1}(x - 2\phi\hat{x}) + \widetilde{I}^{l-1}(x + 2\phi\hat{x})) + 4\eta\alpha x.$$

Here, $\eta = 1/\ell$ is an error parameter. This equation gives the flattening from $l - 1$ to l. Since $\widetilde{I}^{l-1}$ is concave, the averaging in the first part shows that $\widetilde{I}^l(x)$ is much smaller than $\widetilde{I}^{l-1}(x)$. Note that additive error term, which does not occur in [15]. This shows that when x is large, this bound is not interesting. That is no surprise, because we can only sample some prefix of I^l. Then, we prove by induction on l that, if we define

$$\psi = -\log\left(\frac{1}{2}(\sqrt{1 - 2\phi} + \sqrt{1 + 2\phi})\right) = \zeta\tau,$$

then

$$\widetilde{I}^l(x) \leqslant e^{3\eta l}\left[\sqrt{x}e^{-\psi l} + \frac{x}{2m}\right] + 4e^{4\eta l}\alpha x.$$

Recall that $\eta = 1/\ell$. The $e^{-\psi l}$ term decays very rapidly. For the final $\ell \geqslant \log(m)/\zeta$, and for $x = 1$, all terms become $O(\alpha)$. We then get

$$\max_j \frac{\widetilde{p}_j^\ell}{2d_j} = \widetilde{I}^\ell(1) \leqslant O(e^{-\psi\ell} + 1/m + \alpha) \leqslant O(\alpha).$$

4.1 Detailed proof of Theorem 4.1

First, we note that $\phi \leqslant \sqrt{2\zeta\tau}$, so $1 - 2\phi > 0$. Consider the CUTORBOUND algorithm. It is easy to see that this algorithm can be implemented to run in time $O(\Delta \log^4(n)/\alpha)$, because $w = O(\log^3(n)/\alpha)$, $b = O(\log(n)/\alpha)$ and $\ell = O(\log(n))$.

We now prove that this algorithm has the claimed behavior. We make use of the Lovász-Simonovits curve technique. For every length $l = 0, 1, \ldots, \ell$, let p^l be the probability vector induced on vertices after running a random walk of length l.

Now, we construct the Lovász-Simonovits curve [15], $I^l : [0, 2m] \to [0, 1]$ as follows. Let $j_1, j_2, \ldots, j_n$ be an ordering of the vertices as follows:

$$\frac{p_{j_1}^l}{\omega(j_1)} \geqslant \frac{p_{j_2}^l}{\omega(j_2)} \geqslant \cdots \geqslant \frac{p_{j_1}^l}{\omega(j_n)}.$$

For $k \in \{1, \ldots, n\}$, define the set $S_k^l = \{j_1, j_2, \ldots, j_k\}$. For convenience, we define $S_0^l = \emptyset$, the empty set. For a subset of vertices S, and a probability vector p, define $p(S) = \sum_{i \in S} p_i$. Then, we define the curve I^l at the following points: $I^l(\omega(S_k^l)) := p^l(S_k^l)$, for $k = 0, 1, 2, \ldots, n$. Now we complete the curve I^l by interpolating between these points using line segments. Note that the slope of the line segment of the curve at the points $\omega(S_k^l), \omega(S_{k+1}^l)$ is exactly $\frac{p_{j_{k+1}}^l}{\omega(j_{k+1})}$. A direct definition of the curve is the following: for any point $x \in [0, 2m]$, if k is the unique index where $x \in [\omega(S_k^l), \omega(S_{k+1}^l))$, then $I^l(x) = p^l(S_k^l) + (x - \omega(S_k^l)) \cdot \frac{p_{j_{k+1}}^l}{\omega(j_{k+1})}$.

An useful alternative definition for $I^l(x)$ is the following:

$$I^l(x) = \max \sum_i p_i^l w_i$$

s.t. $w_1, w_2, \ldots, w_n \in [0, 1]; \sum_i \omega(i)w_i \leqslant x.$ (2)

Note that this curve is concave because the slopes of the line segments are decreasing. Also, it is an increasing function. Now, Lovász and Simonovits prove the following facts about the curve: let $S \subseteq V$ be any set of vertices, and let $x_S = \omega(S)$ and ϕ_S be its conductance. For $x \in [0, 2m]$, define $\hat{x} = \min\{x, 2m - x\}$. Then, we have the following:

$$p^l(S) \leqslant \frac{1}{2}(I^{l-1}(x_S - 2\phi_S\hat{x}_S) + I^{l-1}(x_S + 2\phi_S\hat{x}_S)).$$ (3)

Furthermore, for any $x \in [0, 2m]$, we have $I^l(x) \leqslant I^{l-1}(x)$.

The issue now is that it is not possible to compute the p_j^l's exactly since we only use random walks. Fix an error parameter $\eta = 1/\ell$. In the algorithm CUTORBOUND, we run $w = c \cdot \frac{1}{\alpha} \cdot \ln(n)$ walks of length ℓ, where $c = 30/\eta^2$. For each length l, $0 \leqslant l \leqslant \ell$, consider the empirical distribution $\widetilde{p}^l$ induced by the walks on the vertices of the graph, i.e. $\widetilde{p}_j^l = w_j/w$, where w_j is the number of walks of length l ending at j. We search for low conductance cuts by ordering the vertices in decreasing order of $\widetilde{p}^l$ and checking the sets of top k vertices in this order, for all $k = 1, 2, \ldots, O(1/\eta\alpha)$. This takes time $O(w\ell)$. To show that this works, first, define $\widetilde{I}^l$ be the Lovász-Simonovits curve corresponding to $\widetilde{p}^l$. Then, we have the following:

Lemma 4.2 *With probability $1 - o(1)$, the following holds. For every vertex subset of vertices $S \subseteq V$, we*

have

$$(1-\eta)p^l(S)-\eta\alpha\omega(S) \;\leqslant\; \widetilde{p}^l_j \;\leqslant\; (1+\eta)p^l(S)+\eta\alpha\omega(S).$$

For every length l, and every $x \in [0, 2m]$,

$$(1 - \eta)I^l(x) - \eta\alpha x \;\leqslant\; \tilde{I}^l(x) \;\leqslant\; (1 + \eta)I^l(x) + \eta\alpha x.$$

PROOF For any vertex j, define $\eta_j = \eta(p^l_j + \alpha)$. By Bernstein's inequality, we have

$$\begin{aligned}\mathbf{Pr}[|\widetilde{p}^l_j - p^l_j| > \eta_j] &\leqslant 2\exp\left(-\frac{\eta_j^2 w}{2p^l_j + 2\eta_j/3}\right) \\ &< 2\exp(-\eta^2 c\ln(n)/3) \\ &\leqslant 1/n^{10}\end{aligned}$$

since $c = 30/\eta^2$. So with probability at least $1 - o(1)$, for all lengths l, and for all vertices j, we have

$$(1 - \eta)p^l_j - \eta\alpha \;\leqslant\; \widetilde{p}^l_j \;\leqslant\; (1 + \eta)p^l_j + \eta\alpha.$$

Assume this is the case. This immediately implies that for any set S, we have

$$(1-\eta)p^l(S)-\eta\alpha|S| \;\leqslant\; \widetilde{p}^l(S) \;\leqslant\; (1+\eta)p^l(S)+\eta\alpha|S|.$$

Now, because both curves I^l and $\tilde{I}^l$ are piecewise linear, concave and increasing, to prove the lower bound in the claimed inequality, it suffices to prove it for only $x = x_k = \omega(S^l_k)$, for $k = 0, 1, \dots, n$. So fix such an index k.

Now, $I^l(x_k) = p^l(S^l_k)$. Consider $\widetilde{p}^l(S^l_k)$. We have

$$\begin{aligned}\widetilde{p}^l(S^l_k) &\geqslant (1 - \eta)p^l(S^l_k) - \eta\alpha|S^l_k| \\ &\geqslant (1 - \eta)p^l(S^l_k) - \eta\alpha\omega(S^l_k).\end{aligned}$$

Now, the alternative definition of the Lovász-Simonovits curve (2) implies that $\tilde{I}^l(\omega(S^l_k)) \geqslant \widetilde{p}^l(S^l_k)$, so we get

$$\tilde{I}^l(x_k) \;\geqslant\; (1 - \eta)p^l(S^l_k) - \eta\alpha x_k,$$

as required. The upper bound is proved similarly, considering instead the corresponding sets $\tilde{S}^l_k$ for $\tilde{I}^l$ consisting of the top k vertices in $\widetilde{p}^l$ probability. $\qquad\square$

The algorithm CUTORBOUND can be seen to be searching for low conductance cuts in the top b vertices in the order given by $\widetilde{p}^l_j/\omega(j)$. Now, we prove that if we only find large conductance cuts, then the curve $\tilde{I}^l$ "flattens" out rapidly. Let $j'_1, j'_2, \dots, j'_n$ be this order. Let $\tilde{S}^l_k = \{j'_1, j'_2, \dots, j'_k\}$ be the set of top k vertices in the order, $x_k = \omega(\tilde{S}^l_k)$, and ϕ_k be the conductance of $\tilde{S}^l_k$. Now we are ready to show our flattening lemma:

Lemma 4.3 *With probability $1 - o(1)$, the following holds. Suppose the algorithm* CUTORBOUND *finds only cuts of conductance ϕ when sweeping over the top b vertices in $\widetilde{p}^l$ probability. Then, for any index $k = 0, 1, \dots, n$, we have*

$$p^l(\tilde{S}^l_k) \leqslant \frac{1}{2}(I^{l-1}(x_k-2\phi\hat{x}_k)+I^{l-1}(x_k+2\phi\hat{x}_k))+\eta\alpha\phi\hat{x}_k.$$

PROOF Let $G = \left\{j : \frac{p^{l-1}_j}{\omega(j)} > \eta\alpha\right\}$. We have

$$1 \geqslant p^{l-1}(G) > \eta\alpha\omega(G),$$

so $\omega(G) < 1/\eta\alpha$.

As defined in the algorithm CUTORBOUND, let $b = \lceil\frac{1}{2(1-2\phi)\eta\alpha}\rceil$. Let a be the largest index so that $\widetilde{p}^l_{j'_a} > 0$. If $a < b$, then let Z be the set of $b-a$ vertices k of zero $\widetilde{p}^l$ probability considered by algorithm CUTORBOUND for searching for low conductance cuts. We assume that in choosing the ordering of vertices to construct $\tilde{I}^l$, the vertices in Z appear right after the vertex j'_a. This doesn't change the curve $\tilde{I}^l$ since the zero $\widetilde{p}^l$ probability vertices may be arbitrarily ordered.

Suppose that the algorithm CUTORBOUND finds only cuts of conductance at least ϕ when running over the top b vertices. Then, let k be some index in $0, 1, \dots, n$. We consider two cases for the index k:

Case 1: $k \leqslant b$:
In this case, since the sweep only yielded cuts of conductance at least ϕ, we have $\phi_k \geqslant \phi$. Then (3) implies that

$$p^l(\tilde{S}^l_k) \;\leqslant\; \frac{1}{2}(I^{l-1}(x_k - 2\phi\hat{x}_k) + I^{l-1}(x_k + 2\phi\hat{x}_k)).$$

Case 2: $k > b$:
We have

$$x_k > x_b = \omega(\tilde{S}^l_b) \geqslant 2b \geqslant \frac{1}{(1-2\phi)\eta\alpha} > \frac{1}{1-2\phi}\omega(G).$$

Thus, $\omega(G) < (1-2\phi)x_k \leqslant x_k - 2\phi\hat{x}_k$. Hence, the slope of the curve I^{l-1} at the point $x_k - 2\phi\hat{x}_k$ is at most $\eta\alpha$. Since the curve I^{l-1} is concave and increasing, we conclude that

$$I^{l-1}(x_k - 2\phi\hat{x}_k) \;\geqslant\; I^{l-1}(x_k) - 2\eta\alpha\phi\hat{x}_k,$$

and

$$I^{l-1}(x_k + 2\phi\hat{x}_k) \;\geqslant\; I^{t-1}(x_k).$$

Since $p^l(\tilde{S}^l_k) \leqslant I^l(x_k) \leqslant I^{l-1}(x_k)$,

$$p^l(\tilde{S}^l_k) \leqslant \frac{1}{2}(I^{l-1}(x_k-2\phi\hat{x}_k)+I^{l-1}(x_k+2\phi\hat{x}_k))+\eta\alpha\phi\hat{x}_k.$$

This completes the proof of the lemma. $\qquad\square$

Since the bounds of Lemma 4.2 hold with probability $1 - o(1)$, we assume from now on that is indeed the case for all lengths l. Thus, we conclude that if we never find a cut of conductance at most ϕ, and for any index $k = 0, 1, \ldots, \ell$, we have

$$
\begin{aligned}
\widetilde{I}_k^l(x_k) \\
&= \widetilde{p}_k^l(\widetilde{S}_k^l) \\
&\leqslant (1+\eta)p_k^l(\widetilde{S}_k^l) + \eta\alpha x_k \quad \text{(by Lemma 4.2)} \\
&\leqslant \frac{1+\eta}{2}(I^{l-1}(x_k - 2\phi\hat{x}_k) + I^{l-1}(x_k + 2\phi\hat{x}_k)) \\
&\qquad + 2\eta\alpha x_k \qquad\qquad \text{(by Lemma 4.3)} \\
&\leqslant \frac{1+\eta}{2(1-\eta)}(\widetilde{I}^{l-1}(x_k - 2\phi\hat{x}_k) + \widetilde{I}^{l-1}(x_k + 2\phi\hat{x}_k)) \\
&\qquad + 4\eta\alpha x_k \qquad\qquad \text{(by Lemma 4.2)}
\end{aligned}
$$

Here, we use the facts that $(1+\eta)\phi \leqslant 1$, and $\frac{1+\eta}{1-\eta} \leqslant 2$. Now, because $\widetilde{I}^l$ is a piecewise linear and concave function, where the slope only changes at the x_k points, the above inequality implies that for all $x \in [0, 2m]$, we have

$$
\widetilde{I}^l(x) \leqslant \frac{e^{3\eta}}{2}(\widetilde{I}^{l-1}(x - 2\phi\hat{x}) + \widetilde{I}^{l-1}(x + 2\phi\hat{x})) + 4\eta\alpha x. \tag{4}
$$

Here, we used the bound $\frac{1+\eta}{1-\eta} \leqslant e^3\eta$.

Now, assume that we never find a cut of conductance at most ϕ over all lengths l. Define

$$
\psi = -\log\left(\frac{1}{2}(\sqrt{1-2\phi} + \sqrt{1+2\phi})\right) = \zeta\tau.
$$

Note that $\psi \geqslant \phi^2/2$. Then, we prove by induction on l that

$$
\widetilde{I}^l(x) \leqslant e^{3\eta l}\left[\sqrt{\hat{x}}e^{-\psi l} + \frac{x}{2m}\right] + 4e^{4\eta l}\alpha x.
$$

The statement for $l = 0$ is easy to see, since the curve $I^0(x) = \min\{x/2d_i, 1\}$ (recall that we start the walk at vertex i). Assuming the truth of this bound for $l-1$, we now show it for l. We have

$$
\widetilde{I}^l(x)
$$

$$
\leqslant \frac{e^{3\eta}}{2}(\widetilde{I}^{l-1}(x - 2\phi\hat{x}) + \widetilde{I}^{l-1}(x + 2\phi\hat{x})) + 4\eta\alpha x \tag{5}
$$

$$
\leqslant \frac{e^{3\eta l - \psi(l-1)}}{2}\left[\sqrt{(x - 2\phi\hat{x})} + \sqrt{(x + 2\phi\hat{x})}\right]
$$

$$
+ e^{3\eta l} \cdot \frac{x}{2m} + e^{3\eta} \cdot 4e^{4\eta(l-1)}\alpha x + 4\eta\alpha x \tag{6}
$$

$$
\leqslant e^{3\eta l}\left[\sqrt{\hat{x}}e^{-\psi l} + \frac{x}{2m}\right] + 4e^{4\eta l}\alpha x, \tag{7}
$$

which completes the induction. Here, inequality (5) follows from (4), inequality (6) is by the induction hypothesis, and inequality (7) is based on the following bounds: if $x \leqslant m$, then

$$
\begin{aligned}
\sqrt{(x - 2\phi\hat{x})} + \sqrt{(x + 2\phi\hat{x})} &\leqslant \sqrt{x - 2\phi x} + \sqrt{x + 2\phi x} \\
&= 2\hat{x}e^{-\psi},
\end{aligned}
$$

and if $x > m$, then

$$
\begin{aligned}
\sqrt{(x - 2\phi\hat{x})} + \sqrt{(x + 2\phi\hat{x})} &\leqslant \sqrt{2m - (x - 2\phi(2m - x))} \\
&\quad + \sqrt{2m - (x + 2\phi(2m - x))} \\
&= 2\hat{x}e^{-\psi}.
\end{aligned}
$$

Next, since $\eta = 1/\ell$, we get

$$
\max_j \frac{\widetilde{p}_j^\ell}{2d_j} = \widetilde{I}^\ell(1) \leqslant e^{-\psi\ell+3} + \frac{e^3}{2m} + 4e^4\alpha \leqslant 250\alpha,
$$

assuming $\alpha = m^{-\tau}$, $\ell = \frac{\ln m}{\zeta}$, and $\psi = \zeta\tau$. Finally, again invoking Lemma 4.2, we get that $\max p_j^l/2d_j \leqslant 256\alpha$, since $\eta = 1/\ell$.

5 Recursive partitioning

Given the procedure FIND-THRESHOLD, one can construct a recursive partitioning algorithm to approximate the MAXCUT. We classify some vertices through FIND-THRESHOLD, remove them, and recurse on the rest of the graph. We call this algorithm SIMPLE. The algorithm BALANCE uses the low conductance sets obtained from Theorem 4.1 and does a careful balancing of parameters to get an improved running time. All proofs of this section, including theoretical guarantees on approximation factors, are in Section 5.1. We state the procedure SIMPLE first and provide the relevant claims.

SIMPLE
Input: Graph G.
Parameters: ε, μ.
1. If $f(\sigma(\varepsilon, \mu)) = 1/2$, then put each vertex in L or R uniformly at random (and return).
2. Let P be a set of $O(\log n)$ vertices chosen uniformly at random.

(a) For all $i \in P$, run procedures FIND-THRESHOLD(i, μ) in parallel. Stop when any one of these succeeds or all of them fail.

3. If all procedures failed, output FAIL.

4. Otherwise, let the successful output be the set $\mathcal{E}_i$ and $\mathcal{O}_i$. With probability $1/2$, put $\mathcal{E}_i$ in L and $\mathcal{O}_i$ in R. With probability $1/2$, do the opposite.

5. Let $\xi = 1 - \mathrm{Inc}(\mathcal{E}_i, \mathcal{O}_i)/m$. Set $\varepsilon' = \varepsilon/\xi$ and G' be the induced subgraph on unclassified vertices. Run SIMPLE(G', ε', μ). If it succeeds, output SUCCESS and return the final cut L and R. If it fails, produce a random cut and output FAIL.

BALANCE
Input: Graph G.
Parameters: $\varepsilon_1, \mu_1, \mu_2, \tau$.

1. Define $\alpha = m^{-\tau}$, $\zeta = 2(\varepsilon_1 + \delta)/\mu_1$.

2. Let P be a random subset of $O(\log n)$ vertices.

3. For each vertex $i \in P$, run CUTORBOUND $(i, \tau, \zeta, \ell(\varepsilon_1, \mu_1))$.

4. If a low conductance set S was found by any of the above calls:

   (a) Let G_S be the induced graph on S, and G' be the induced graph on $V \setminus S$. Run SIMPLE$'(G_S, \mu_2)$ and BALANCE$(G', \varepsilon_1, \mu_1, \mu_2, \tau)$ to get the final partition.

5. Run SIMPLE$(G, \varepsilon_1, \mu_1)$ up to Step 4, using random vertex set P. Then run BALANCE$(G', \varepsilon_1, \mu_1, \mu_2, \tau)$, where G' is the induced graph on the unclassified vertices.

6. Output the better of this cut and a random cut.

The guarantees of SIMPLE are in terms of a function $H(\varepsilon, \mu)$, defined below.

Definition 5.1 *[$H(\varepsilon, \mu)$.] For a given ε and μ, let*

$$z^* = \max\{z : f(\sigma(\varepsilon/z, \mu)) = 1/2\}.$$

Then

$$H(\varepsilon, \mu) := z^*/2 + \int_{z=z^*}^{1} f(\sigma(\varepsilon/z, \mu)) dz.$$

Lemma 5.2 *Let* $\mathrm{MAXCUT}(G) = 1 - \varepsilon$. *There is an algorithm* SIMPLESEARCH(G, μ) *that, with high probability, outputs a cut of value* $H(\varepsilon, \mu) - o(1)$, *and thus the worst-case approximation ratio is* $\min_\varepsilon \frac{H(\varepsilon, \mu)}{1-\varepsilon} - o(1)$. *The running time is* $\widetilde{O}(\Delta m^{2+\mu})$.

The algorithm SIMPLESEARCH is a version of SIMPLE that only takes μ as a parameter and searches for the appropriate value of ε. The procedure SIMPLESEARCH runs SIMPLE(G, ε_r, μ), for all ε_r such that $1 - \varepsilon_r = (1-\gamma)^r$ and $1/2 \leqslant 1 - \varepsilon_r \leqslant 1$, and returns the best cut found. By choosing γ small enough and Claim 5.3 below, we can ensure that if $\mathrm{MAXCUT}(G) = 1 - \varepsilon$, then SIMPLESEARCH$(G, \mu)$ returns a cut of value least $H(\varepsilon, \mu) - o(1)$. It therefore suffices to prove:

Claim 5.3 *If* SIMPLE(G, ε, μ) *succeeds, it outputs a cut of value at least* $H(\varepsilon, \mu)$. *If it fails, it outputs a cut of value* $1/2$. *If* $\mathrm{MAXCUT}(G) \geqslant 1 - \varepsilon$, *then* SIMPLE$(G, \varepsilon, \mu)$ *succeeds with high probability. The running time is always bounded by* $\widetilde{O}(\Delta m^{2+\mu})$.

We now describe BALANCE and state the main lemma associated with it[8]. We observe that BALANCE uses CUTORBOUND to either decompose the graph into pieces, or ensure that we classify many vertices. We use Theorem 4.1 to bound the running time.

Lemma 5.4 *For any constant* $b > 1.5$, *there is a choice of* ε_1, μ_1, μ_2 *and* τ *so that* BALANCE *runs in* $\widetilde{O}(\Delta m^b)$ *time and provides an approximation factor that is a constant greater than* 0.5.

Let us give an intuitive explanation for the 1.5-factor in the exponent for the running time. Neglecting the μ's and polylogarithmic factors, we perform $O(1/\alpha)$ walks in CUTORBOUND. In the worst case, we could get a low conductance set of constant size, in which case the work per output is $O(1/\alpha)$. When we have the α bound on probabilities, the work per output is $O(\alpha m)$. So it appears that $\alpha = 1/\sqrt{m}$ is the balancing point, which yields an $\widetilde{O}(m^{1.5})$ time algorithm.

In the next subsection, we define many parameters which will be central to our analysis. We then provide detailed proofs for Claim 5.3 and Lemma 5.4. Finally, we give a plot detailing how the approximation factor increases with running time (for both SIMPLE and BALANCE).

---

[8]We denote its parameter as ε_1 since we will use the variable ε to denote other quantities.

5.1 Preliminaries

For convenience, we list the various free parameters and dependent variables.

- ε is the maxcut parameter, as described above. Eventually, this will be set to some constant (this is explained in more detail later).

- μ is a running time parameter. This is used to control the norm of the $\widetilde{y}_i$ vector, and through that, the running time. This affects the approximation factor obtained, through Lemma 3.7.

- $\alpha(= m^{-\tau})$ is the maximum probability parameter. This directly affects the running time through Lemma 3.5. For SIMPLE, this is just set to 1, so it only plays a role in BALANCE.

- $\ell(\varepsilon, \mu) := \mu(\ln(4m/\delta^2)/[2(\delta + \varepsilon)]$. This is the length of the random walk.

- $\sigma(\varepsilon, \mu)$ is the parameter that is in Lemma 3.5. Setting $\varepsilon' = -\ln(1 - \varepsilon)/\mu$, we get $1 - \sigma = e^{-\varepsilon'}(1 - \varepsilon)(1 - \delta)(1 - \gamma)$.

- $\chi(\varepsilon, \mu, \alpha)$ is the cut parameter that comes from Theorem 4.1. When we get a set S of low conductance, the number of edges in the cut is at most $\chi(\varepsilon, \mu)|\texttt{Internal}(S)|$. Here, $\texttt{Internal}(S)$ is the set of edges internal to S. In Theorem 4.1, the number of cut edges in stated in terms of the conductance ϕ. We have $\chi = 4\phi/(1 - 2\phi)$. Also, ϕ is at most $\sqrt{4\varepsilon\tau/\mu}$. We will drop the dependence on α, since it will be fixed (more details given later).

We will also use some properties of the function $H(\varepsilon, \mu)$.

Lemma 5.5 *For any fixed $\mu > 0$, $H(\varepsilon, \mu)$ is a convex, decreasing function of ε. Furthermore, there is a value $\bar{\varepsilon} = \bar{\varepsilon}(\mu)$ such that $H(\bar{\varepsilon}, \mu) > 0.5029$.*

PROOF First, note that $f(\sigma)$ is a decreasing function of σ. This is because all the three functions that define f are decreasing in their respective ranges, and the transition from one function to the next occurs precisely at the point where the functions are equal.

Now, for any fixed μ, $\sigma(\varepsilon, \mu)$ is a strictly increasing function of ε, and hence, $f(\sigma(\varepsilon, \mu))$ is a decreasing function of ε. Thus, $H(\varepsilon, \mu) = \int_0^1 f(\sigma(\varepsilon/r, \mu))dr$ is a decreasing function of ε, since for any fixed r, the integrand $f(\sigma(\varepsilon/r, \mu))$ is a decreasing function of ε.

For convenience of notation, we will use H and σ to refer $H(\varepsilon, \mu)$ and $\sigma(\varepsilon, \mu)$ respectively. Now define $x = \varepsilon/r$. Doing this change of variables in the integral,

we get $H = \varepsilon \int_\varepsilon^\infty \frac{f(\sigma(x, \mu))}{x^2} dx$. By the fundamental theorem of calculus, we get that

$$\frac{\partial H}{\partial \varepsilon} = \int_\varepsilon^\infty \frac{f(\sigma(x, \mu))}{x^2} dx - \frac{f(\sigma)}{\varepsilon}.$$

Again applying the fundamental theorem of calculus, we get that

$$\frac{\partial^2 H}{\partial \varepsilon^2} = -\frac{f(\sigma)}{\varepsilon^2} - \frac{\varepsilon\frac{\partial f(\sigma)}{\partial \varepsilon} - f(\sigma)}{\varepsilon^2} = -\frac{1}{\varepsilon} \cdot \frac{\partial f(\sigma)}{\partial \varepsilon} \geqslant 0,$$

since $f(\sigma)$ is a decreasing function of ε. Thus, H is a convex function of ε.

To show the last part, let σ_μ^{-1} is the inverse function of $\sigma(\varepsilon, \mu)$, keeping μ fixed, and consider $\bar{\varepsilon}(\mu) = \sigma_\mu^{-1}(1/4) = 1 - (\frac{3}{4})^{\frac{\mu}{1+\mu}} - o(1)$, by making δ and γ small enough constants. For $r \in [1/4, 1/3]$, we have $f(\sigma(\bar{\varepsilon}/r, \mu)) \geqslant f(1/4) > 0.535$. Thus, we get

$$H(\bar{\varepsilon}, \mu) > 0.5 + 0.035 \times (1/3 - 1/4) = 0.5029.$$

$\square$

5.2 Proof for SIMPLE

As we showed in the main body, it suffices to prove Claim 5.3.

PROOF (of Claim 5.3) This closely follows the analysis given in [24] and [19]. If any recursive call to SIMPLE fails, then the top level algorithm also fails and outputs a random cut.

Suppose $\text{MAXCUT}(G)$ is at least $1 - \varepsilon$. Then $MaxCut(G')$ is at least

$$\frac{(1 - \varepsilon)m - \texttt{Inc}(\mathcal{E}_i, \mathcal{O}_i)}{m - \texttt{Inc}(\mathcal{E}_i, \mathcal{O}_i)} = 1 - \varepsilon/\xi$$

Applying this inductively, we can argue that *whenever* a recursive call $\text{SIMPLE}(G', \varepsilon', \mu)$ is made, $MaxCut(G') \geqslant 1 - \varepsilon'$. From Lemma 3.7, since $O(\log n)$ vertices are chosen in P, with high probability, in *every* recursive call, a good vertex is present in P. From Lemma 3.5, in every recursive call, with high probability, some call to FIND-THRESHOLD succeeds. Hence, SIMPLE will not output FAIL and succeeds.

Assuming the success of SIMPLE, let us compute the total number of edges cut. We denote the parameters of the tth recursive call to SIMPLE by subscripts of t. Let the number of edges in G_t be $\rho_t m$ (where $\rho_0 = 1$). Let T be the last call to SIMPLE. We have $\varepsilon_t = \varepsilon/\rho_t$.

Only for $t = T$, we have that $f(\sigma(\varepsilon/\rho_t, \mu)) = 1/2$. Let z^* be defined according to Definition 5.1. Note that $\rho_t \leqslant z^*$. In the last round, we cut $\rho_T m/2$ edges. The number of cut edges in other rounds is $f(\sigma(\varepsilon_t, \mu))(\rho_t - \rho_{t+1})m$. Summing over all t, the total number of edges cut (as a fraction of m) is

$$\sum_{t=0}^{T-1} f(\sigma(\varepsilon_t, \mu))(\rho_t - \rho_{t+1}) + \rho_T/2$$
$$= \sum_{t=0}^{T-2} \int_{\rho_{t+1}}^{\rho_t} f(\sigma(\varepsilon/\rho_t, \mu))dr + \int_{\rho_T}^{\rho_{T-1}} f(\sigma(\varepsilon/\rho_t, \mu))dr$$
$$\quad + \rho_T/2$$
$$= \sum_{t=0}^{T-2} \int_{\rho_{t+1}}^{\rho_t} f(\sigma(\varepsilon/\rho_t, \mu))dr + \int_{\rho_T}^{z^*} f(\sigma(\varepsilon/\rho_t, \mu))dr$$
$$\quad + \int_{z^*}^{\rho_{T-1}} f(\sigma(\varepsilon/\rho_t, \mu))dr + \rho_T/2.$$

To bound this from below, we need a few observations. First, note that σ is an increasing function of its first argument. Hence, $\sigma(\varepsilon/r, \mu)$ is a decreasing function of r. Since f is a decreasing function (of its single argument), $f(\sigma(\varepsilon/r, \mu))$ is an increasing function of r. So, for $r \leqslant \rho_t$, $f(\sigma(\varepsilon/\rho_t, \mu)) \geqslant f(\sigma(\varepsilon/r, \mu))$. We have $\rho_T \leqslant z^*$. The worst case for us (when the minimum number of edges is cut) is when ρ_T is exactly z^*. This is because we cut the lowest fraction $(1/2)$ of edges for G_T. So, we get

$$\sum_{t=0}^{T-1} f(\sigma(\varepsilon_t, \mu))(\rho_t - \rho_{t+1}) + \rho_T/2$$
$$\geqslant \sum_{t=0}^{T-2} \int_{\rho_{t+1}}^{\rho_t} f(\sigma(\varepsilon/r, \mu))dr + \int_{z^*}^{\rho_{T-1}} f(\sigma(\varepsilon/r, \mu))dr$$
$$\quad + z^*/2$$
$$= \int_{z^*}^{1} f(\sigma(\varepsilon/r, \mu))dr + z^*/2.$$

We now bound the running time, using Lemma 3.5. Consider a successful iteration t. Suppose the number of vertices classified in this iteration is N_t. The total running time in iteration t is $\widetilde{O}(N_t \Delta m^{1+\mu})$. This is because we run the $O(\log n)$ calls in parallel, so the running time is at most $O(\log n)$ times the running time of the successful call. Summed over all iterations, this is at most $\widetilde{O}(\Delta m^{2+\mu})$. Suppose an iteration is unsuccessful, the total running time is $\widetilde{O}(\Delta m^{2+\mu})$. There can only be one such iteration, and the claimed bound follows. $\square$

5.3 Proofs for BALANCE

We first give a rather complicated expression for the approximation ratio of BALANCE. First, for any $\mu > 0$, define $h(\mu) = \min_\varepsilon \frac{H(\varepsilon, \mu)}{1 - \varepsilon}$. This is essentially the approximation factor of SIMPLESEARCH.

Claim 5.6 *The algorithm* BALANCE *has a work to output ratio of* $\widetilde{O}(\Delta(m^{\tau + \mu_2 \tau} + m^{1 + \mu_1 - \tau}))$. *The approximation ratio is at least*

$$\max_{\varepsilon_1} \min \left\{ \mathcal{H}(\varepsilon_1, \mu_1, \mu_2), H(\varepsilon_1, \mu_1), \frac{1}{2(1 - \varepsilon_1)} \right\},$$

where $\mathcal{H}(\varepsilon_1, \mu_1, \mu_2) :=$

$$\min_\varepsilon \max \left\{ \frac{1}{2(1-\varepsilon)}, \frac{h(\mu_2)(1 - \varepsilon - \varepsilon\chi(\varepsilon_1, \mu_1)) + \chi(\varepsilon_1, \mu_1)/2}{(1-\varepsilon)(1 + \chi(\varepsilon_1, \mu_1))} \right\}.$$

PROOF: First let us analyze the work per output ratio of BALANCE. We initially perform $\widetilde{O}(\Delta m^\tau)$ walks. Suppose we get a low conductance set S. We then run SIMPLE$(G_S, \varepsilon_2, \mu_2)$. Here, the work to output ratio is at most $\widetilde{O}(\Delta m^{\tau + \mu_2 \tau})$. If we get a tripartition, the work to output ratio is at most $\widetilde{O}(\Delta m^{1 + \mu_1 - \tau})$. Adding these, we get an upper bound on the total work to output ratio.

Because we choose a random subset P of size $O(\log n)$, we will assume that Lemma 5.2 and Claim 5.3 hold (without any error). To analyze the approximation ratio, we follow the progress of the algorithm to the end. In each iteration, either a low conductance set is removed, or the basic algorithm is run. In each iteration, let us consider the set of vertices this is assigned to some side of the final cut. In case of a low conductance set, we get a cut for the whole set. Otherwise, if we get a tripartition, the union $\mathcal{E}_i \cup \mathcal{O}_i$ will be this set. If we do not get a tripartition, then we output a random cut (thereby classifying all remaining vertices). Let us number the low conductance sets as $S_1, S_2, \ldots$. The others are denoted $T_1, T_2, \ldots, T_f$. We will partition the edges of G into parts, defining subgraphs. The subgraph G_S consists of all edges incident to some S_i. The remaining edges form G_T. The edges of G_S are further partitioned into two sets: G_c is the subgraph of *cross* edges, which have only one endpoint in S. The other edges make the subgraph G'_S. The edge sets of these subgraphs are E_S, E_T, E_c, E'_S, respectively. For any set S_i, $G|_{S_i}$ denotes the induced subgraph on S_i.

We now count the number of edges in each set that our algorithm cuts. We can only guarantee that half the edges in E_c are cut. Our algorithm will cut (in each S_i) at least $h(\mu_2)$MAXCUT$(G|_{S_i})$ edges. This

deals with all the edges in E_S. Let the MAXCUT value of the subgraph G_S be $1 - \varepsilon$. Then, trivially we cut at least $\text{MAXCUT}(G_S)/[2(1-\varepsilon)]$ edges. In E_{T_f}, we can only cut half of the edges. In E_{T_j}, we cut an $H(\varepsilon_1, \mu_1)$ fraction of edges. In total,

$$\max\left\{\frac{\text{MAXCUT}(G_S)}{2(1-\varepsilon)}, \sum_i h(\mu_2)\text{MAXCUT}(G|_{S_i}) + \frac{|E_c|}{2}\right\}$$
$$+ \sum_j H(\varepsilon_1, \mu_1)|E_{T_j}| + \frac{|E_{T_f}|}{2}.$$

We first deal with the latter part. The maxcut of $G|_{T_F}$ is at most $(1 - \varepsilon_1)$ (otherwise, we would get a tripartition). So we get,

$$\sum_j H(\varepsilon_1, \mu_1)|E_{T_j}| + \frac{|E_{T_f}|}{2}$$
$$\geqslant \sum_j H(\varepsilon_1, \mu_1)|E_{T_j}| + \frac{\text{MAXCUT}(G|_{T_f})|E_{T_f}|}{2(1-\varepsilon_1)}$$
$$\geqslant \min\left(H(\varepsilon_1, \mu_1), \frac{1}{2(1-\varepsilon_1)}\right)\text{MAXCUT}(G_T).$$

We now handle the former part. By definition, $|E_c| \leqslant \chi(\varepsilon_1, \mu_1)|E_S'|$. Fixing the size of $E_c \cup E_S'$, we minimize the number of edges cut by taking this to be equality. . If we remove the edges E_c, we get the subgraph G_S'. The MAXCUT of G_S' is at least

$$1 - \frac{\varepsilon}{1 - \chi(\varepsilon_1, \mu_1)} = \frac{1 - \varepsilon - \chi(\varepsilon_1, \mu_1)}{1 - \chi(\varepsilon_1, \mu_1)}$$

Now, we lower bound the total number of edges in G_S that are cut.

$$\sum_i h(\mu_2)\text{MAXCUT}(G|_{S_i}) + (1/2)|E_c|$$
$$\geqslant h(\mu_2)\sum_i \text{MAXCUT}(G|_{S_i}) + (1/2)|E_c|$$
$$\geqslant h(\mu_2)\text{MAXCUT}(G_S') + (1/2)\chi(\varepsilon_1, \mu_1)|E_S'|$$
$$\geqslant \left[h(\mu_2)\frac{1 - \varepsilon - \chi(\varepsilon_1, \mu_1)}{1 - \chi(\varepsilon_1, \mu_1)} + (1/2)\chi(\varepsilon_1, \mu_1)\right]|E_S'|.$$

By definition of ε,

$$\text{MAXCUT}(G_S) = (1 - \varepsilon)|E_S| =$$
$$(1 - \varepsilon)(1 + \chi(\varepsilon_1, \mu_1))|E_S'|.$$

Substituting this, we get that the total number of edges cut is at least:

$$\text{MAXCUT}(G_S)\mathcal{H}(\varepsilon_1, \mu_1, \mu_2)$$
$$+\text{MAXCUT}(G_T)\min\left(H(\varepsilon_1, \mu_1), \frac{1}{2(1-\varepsilon_1)}\right)$$

Observe that

$$\text{MAXCUT}(G_S) + \text{MAXCUT}(G_T) \geqslant \text{MAXCUT}(G).$$

The parameter ε_1 can be chosen to maximize the approximation ratio. $\qquad\square$

Using this we prove the main lemma about BALANCE (restated here for convenience):

Lemma 5.7 *For any constant $b > 1.5$, there is a choice of ε_1, μ_1, μ_2 and τ so that there is an $\widetilde{O}(\Delta m^b)$ time algorithm with an approximation factor that is a constant greater than 0.5.*

PROOF: The algorithm BALANCE has a work to output ratio of $\widetilde{O}(\Delta(m^{\tau+\mu_2\tau} + m^{1+\mu_1-\tau}))$. We now set μ_1 and μ_2 to be constants so that the work to output ratio is $b - 1$. For this, we set $\tau + \mu_2\tau = 1 + \mu_1 - \tau = b - 1$. Letting $\mu_1 > 0$ be a free parameter, this gives $\tau = 2 + \mu_1 - b$, and $\mu_2 = \frac{2b-\mu_1-3}{2+\mu_1-b}$. Note that since $b > 1.5$, we can choose $\mu_1 > 0$ so that $\tau \geqslant 0$ and $\mu_2 > 0$. By Claim 5.6, we can decide the value of ε_1 as the value maximizing the approximation ratio.

Now, it remains to show that for any choice of $\mu_1, \mu_2 > 0$, the bound on the approximation factor given by Claim 5.6 is greater than 0.5. For convenience of notation, we will drop the arguments to functions and use h, H, and χ to refer to $h(\mu_2)$, $H(\varepsilon_1, \mu_1)$, and $\chi(\varepsilon_1, \mu_1)$ respectively. First, note that $h > 0.5$. Let us set $\varepsilon_1 = \bar{\varepsilon}(\mu_1)$ as from the statement of Lemma 5.5. Then $H > 0.5029$, and $\frac{1}{2(1-\varepsilon_1)} > 0.5$ since $\varepsilon_1 > 0$. Furthermore, note that $\min_\varepsilon \max\left\{\frac{1}{2(1-\varepsilon)}, \frac{h(1-\varepsilon-\varepsilon\chi)+\chi/2}{(1-\varepsilon)(1+\chi)}\right\}$ is obtained at $\varepsilon = \frac{2h-1}{2h(1+\chi)}$, and takes the value $\frac{h+h\chi}{1+2h\chi} > 0.5$ since $h > 0.5$. Thus, the minimum of all these three quantities is greater than 0.5, and hence the approximation factor is more than 0.5. $\qquad\square$

Using a more nuanced analysis of the approximation ratio, we can get better bounds. This requires the solving of an optimization problem, as opposed to Claim 5.6. We provided the weaker claim because it is easier to use for Lemma 5.4.

Claim 5.8 *Let us fix μ_1, μ_2. The approximation ratio can be bounded as follows: let ε_S', X, Y, Z be variables and $\varepsilon, \varepsilon_1$ be fixed. First minimize the function:*

$$(H(\varepsilon_S', \mu_2) + \chi(\varepsilon_1, \mu_1)/2)X + H(\varepsilon_1, \mu_1)Y + \frac{Z}{2}$$

with constraints:

$$\varepsilon'_S X + \varepsilon_1 Z \leqslant \varepsilon$$
$$(1 + \chi(\varepsilon_1, \mu_1))X + Y + Z = 1$$
$$0 \leqslant \varepsilon'_S \leqslant 1/2$$
$$0 \leqslant X, Y, Z \leqslant 1$$

Let this value by $OBJ(\varepsilon, \varepsilon_1)$. The approximation ratio is at least

$$\max_{\varepsilon_1} \min_{\varepsilon} \max \left\{ \frac{1}{2(1-\varepsilon)}, \frac{OBJ(\varepsilon, \varepsilon_1)}{1-\varepsilon} \right\}.$$

PROOF: To analyze the approximation ratio, we follow the progress of the algorithm to the end. In each iteration, either a low conductance set is removed, or the basic algorithm is run. In each iteration, let us consider the set of vertices this is assigned to some side of the final cut. In case of a low conductance set, we get a cut for the whole set. Otherwise, if we get a tripartition, the union $V_{i,r}^+ \cup V_{i,r}^-$ will be this set. If we do not get a tripartition, then we output a random cut (thereby classifying all remaining vertices). Let us number the low conductance sets as $S_1, S_2, \ldots$. The others are denoted $T_1, T_2, \ldots, T_f$. We will partition the edges of G into parts, defining subgraphs. The subgraph G_S consists of all edges incident to some S_i. The remaining edges form G_T. The edges of G_S are further partitioned into two sets: G_c is the subgraph of *cross* edges, which have only one endpoint in S. The other edges make the subgraph G'_S. In G_T, let the edges incident to vertices *not* in T_f be be G'_T. The remaining edges form the subgraph G_f. The edge sets of these subgraphs are $E_S, E_T, E_c, E'_S, E_f, E'_T$, respectively. For any set S_i, $G|_{S_i}$ denotes the induced subgraph on S_i.

We now count the number of edges in each set that our algorithm cuts. We can only guarantee that half the edges in E_c are cut. Let the MAXCUT of $G|_{S_i}$ be MAXCUT$(G|_{S_i})$ $(= \tau_i)$. Our algorithm will cut (in each S_i) at least $H(\tau_i, \mu_2)|E_{S_i}|$ edges. This deals with all the edges in E_S. In E_{T_f}, we can only cut half of the edges. In E_{T_j}, we cut an $H(\varepsilon_1, \mu_1)$ fraction of edges. In total, the number of edges cut is at least

$$\sum_i H(\tau_i, \mu_2)|E_{S_i}| + \frac{|E_c|}{2} + \sum_j H(\varepsilon_1, \mu_1)|E_{T_j}| + \frac{|E_{T_f}|}{2}.$$

By convexity of H, we have

$$\sum_i H(\tau_i, \mu_2) \geqslant H(\varepsilon'_S, \mu_2)|E'_S|,$$

where MAXCUT$(G'_S) = 1 - \varepsilon'_S$. Putting it all together, we cut at least

$$H(\varepsilon'_S, \mu_2)|E'_S| + H(\varepsilon_1, \mu_1)|E'_T| + (1/2)|E_f| + (1/2)|E_c|$$

We would like to find out the minimum value this can attain, for a given ε_1. The parameters μ_1, μ_2 are fixed. The maxcut of G_f is at most $(1-\varepsilon_1)$ (otherwise, we would get a tripartition). We have the following constraints:

$$|E_c| \leqslant \chi(\varepsilon_1, \mu_1)|E'_S|$$
$$\varepsilon'_S|E'_S| + \varepsilon_f|E_f| \leqslant \varepsilon m$$
$$|E'_S| + |E'_T| + |E_f| + |E_c| = m$$
$$\varepsilon_1 \leqslant \varepsilon_f \leqslant 1/2$$

For a given size of E'_S, we should maximize E_c to cut the least number of edges. So we can assume that $|E_c| = \chi(\varepsilon_1, \mu_1)|E'_S|$. Let us set $X := |E'_S|/m$, $Y := |E'_T|/m$, and $Z := |E_f|/m$. Consider fixing ε and ε_1. The variables are $\varepsilon'_S, \varepsilon_f, X, Y, Z$. This means the number of edges cut is bounded below by the *minimum* of

$$(H(\varepsilon'_S, \mu_2) + \chi(\varepsilon_1, \mu_1)/2)X + H(\varepsilon_1, \mu_1)Y + \frac{Z}{2}$$

under the constraints:

$$\varepsilon'_S X + \varepsilon_f Z \leqslant \varepsilon$$
$$(1 + \chi(\varepsilon_1, \mu_1))X + Y + Z = 1$$
$$\varepsilon_1 \leqslant \varepsilon_f \leqslant 1/2$$
$$0 \leqslant \varepsilon'_S \leqslant 1/2$$
$$0 \leqslant X, Y, Z \leqslant 1.$$

Let $OBJ(\varepsilon, \varepsilon_1)$ be the minimum value attained. We observe that in the optimal solution, we must have $\varepsilon_f = \varepsilon_1$. Otherwise, note that we can reduce the objective by decreasing ε_f. This is because for a small decrease in ε_f, we can increase Z (and decrease either X or Y). This preserves all the constraints, but decreases the objective. The approximation ratio is then lower bounded by

$$\max_{\varepsilon_1} \min_{\varepsilon} \max \left\{ \frac{1}{2(1-\varepsilon)}, \frac{OBJ(\varepsilon, \varepsilon_1)}{1-\varepsilon} \right\}.$$

$\square$

5.4 Running time/approximation ratio tradeoff

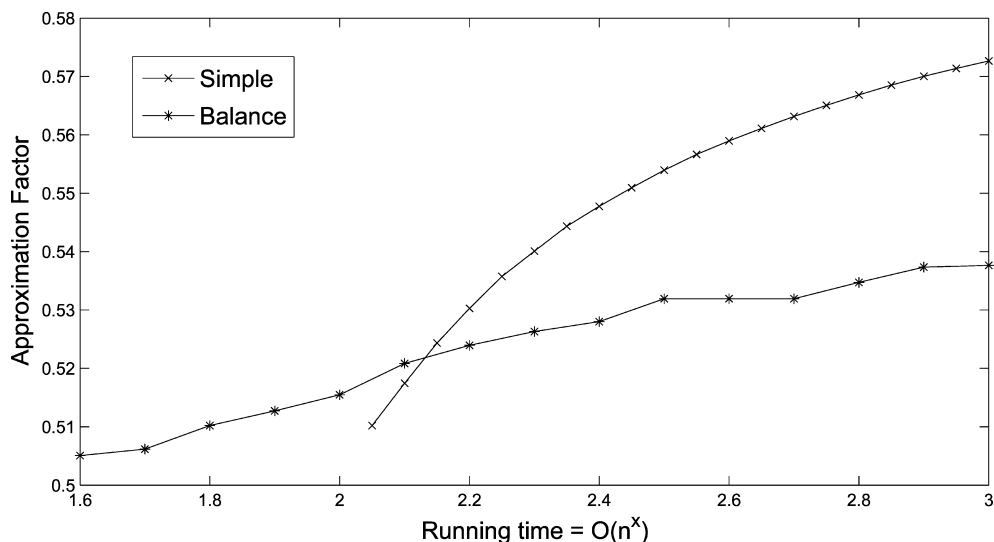Refer to Figure 1 for the tradeoff between running time and approximation factor.

Figure 1: Running time/approximation ratio tradeoff curve for SIMPLE and BALANCE. SIMPLE needs running time $\widetilde{O}(n^{2+\mu})$ and BALANCE needs running time $\widetilde{O}(n^{1.5+\mu})$, for any constant $\mu > 0$. The approximation ratio for SIMPLE is from Lemma 5.2, and that for BALANCE is from Claim 5.8.

6   Conclusions and further work

Our combinatorial algorithm is very natural and simple, and beats the 0.5 barrier for MAXCUT. The current bounds for the approximation ratio we get for, say, quadratic time are quite far from the optimal Goemans-Williamson 0.878, or even from Soto's 0.6142 bound for Trevisan's algorithm. The approximation ratio of our algorithm can probably be improved, and it might be possible to get a better running time. This would probably require newer analyses of Trevisan's algorithm, similar in spirit to Soto's work [19]. It would be interesting to see if some other techniques different from random walks can be used for MAXCUT.

This algorithm naturally suggests whether a similar approach can be used for other 2-CSPs. We believe that this should be possible, and it would provide a nice framework for combinatorial algorithms for such CSPs.

Our local partitioning algorithm raises very interesting questions. Can we get such a partitioning procedure that has a better work to output ratio (close to polylogarithmic) but does not lose the $\sqrt{\log n}$ factor in the conductance (which previous algorithms lose)? We currently have a work to output that can be made close to $\sqrt{n}$ in the worst case. An improvement would be of significant interest.

References

[1] R. Andersen, F. R. K. Chung, and K. Lang.Local graph partitioning using pagerank vectors. In *Proceedings of the Annual 47th Foundations of Computer Science (FOCS)*, pages 475-486, 2006.

[2] R. Andersen and K. Lang.An algorithm for improving graph partitions. In *Proceedings of the 19th Annual Sym-posium of Discrete Algorithms (SODA)*, pages 651-660, 2008.

[3] S. Arora and S. Kale.A combinatorial, primal-dual ap-proach to semide?nite programs. In *Proceedings of the 39th ACM Symposium on Theory of Computing (STOC)*, pages 227-236, 2007.

[4] C. Bazgan and Z. Tuza. Combinatorial 5/6-approximation of max cut in graphs of maximum degree 3. *Journal of Discrete Algorithms,* 6(3):510-519, 2008.

[5] J. A. Bondy and S. C. Locke. Largest bipartite subgraphs in triangle-free graphs with maximum degree three. *Journal of Graph Theory*, 10:477-504, 1986.

[6] M. Datar, T. Feder, A. Gionis, R. Motwani, and R. Pani-grahy. A combinatorial algorithm for max csp. *Informa-tion Processing Letters*, 85(6):307-315, 2003.

[7] W. F. de la Vega and C. Kenyon-Mathieu. Linear program-ming relaxations of maxcut. In *Proceedings of the 18th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 53-61, 2007.

[8] I. Dinur.The PCP theorem by gap amplification. In *Proceedings of the 38th ACM Symposium on Theory of Computing (STOC)*, pages 241-250, 2006.

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

[10] O. Goldreich and D. Ron.A sublinear bipartite tester for bounded degree graphs. *Combinatorica,* 19(3):335-373, 1999.

[11] E. Halperin, D. Livnat, and U. Zwick.Max cut in cubic graphs. *Journal of Algorithms*, 53:169-185, 2004.

[12] R.M. Karp. Reducibility among combinatorial problems. *Complexity of Computer Computations*, pages 85-103, 1972.

[13] S. Khot. On the power of unique 2-prover 1-round games. In *Proceedings of the 34th ACM Symposium on Theory of Computing (STOC),* pages 767-775, 2002.

[14] S. Khot, G. Kindler, E. Mossel, and R. O'Donnell. Optimal inapproximability results for max-cut and other twovariable csps? In *Proceedings of the 45th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 146-154, 2004.

[15] L. Lovász and M. Simonovits. The mixing rate of markov chains, an isoperimetric inequality, and computing the volume. In *FOC*S, pages 346-354, 1990.

[16] M. Mihail. Conductance and convergence of markov chains-a combinatorial treatment of expanders. In *Proceedings of the Annual 30th Foundations of Computer Science (FOCS)*, pages 526-531, 1989.

[17] G. Schoenebeck, L. Trevisan, and M. Tulsiani. Lovaszschrijver lp relaxations of vertex cover and max cut. In *Proceedings of the 39th ACM Symposium on Theory of Computing (STOC)*, pages 302-310, 2007.

[18] A. Sinclair.Improved bounds for mixing rates of markov chains and multicommodity flow. *Combinatorics, Probability & Computing,* 1:351-370, 1992.

[19] J.Soto. Improved analysis of a max cut algorithm based on spectral partitioning. Manuscript at arXiv:0910.0504v1, 2009.

[20] D. Spielman and S.-H. Teng. Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems. In *In Proceedings of the 36th ACM Symposium on Theory of Computing (STOC)*, pages 81-90, 2004.

[21] D. Steurer.Fast sdp algorithms for constraint satisfaction problems. In *Proceedings of the 21st ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 684-697, 2010.

[22] L.Trevisan. Non-approximability results for optimization problems on bounded degree instances. In *Proceedings of the 33rd ACM Symposium on Theory of Computing (STOC)*, pages 453-461, 2001.

[23] L. Trevisan. Approximation algorithms for unique games. In *Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 197-205, 2005.

[24] L. Trevisan. Max cut and the smallest eigenvalue. In *Proceedings of the 41st ACM Symposium on Theory of Computing (STOC)*, pages 263-272, 2009.